MetiLinx

User's Guide
&
Technical Reference

iManager Developer 2.2

APPENDIX

Navigating in iManager

MetiLinx iManager follows standard Windows navigation features, which allow you to carry out the same command in more than one way.

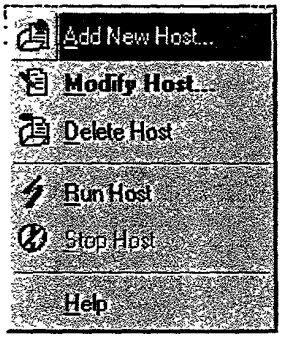
Double-click feature

Use the double-click feature to expand the MetiLinx root and view the MetiLinx tree. Double-click the last subcomponent of a component to carry out the Modify command.

Right-click feature

Use the right-click feature to view a component menu. Right-clicking a component within the MetiLinx tree (control pane) provides a menu with limited list of enabled commands for that component. Right-clicking a component within the details pane (right pane) provides a menu with a complete list of enabled commands for that component. For example, if you right-click Hosts in the control pane, the enable commands that appear are limited to Add New Host and Help. When you right-click on a host system displayed on the right pane, Modify Host, Delete Host, Run Host, and Help are enabled.

Hosts right-click menu from left panel



Host system right-click menu from right panel

Drag and drop feature

Use the iManager drag-and-drop feature to complete host maintenance tasks.

- Add users to a Host System by dragging and dropping User IDs from **Users** into **Host Users.**
- Add Connection Objects to a Host System by dragging and dropping connection objects from the **Object Repository** to **Connections.**

Additional Tools

iAgent is a utility application that starts the iManager components. The iManager installation places a shortcut to the Agent in the Startup group for All Users, so that the agent runs at logon time. By default, the agent starts each host, the Transaction Load Object (TLO)and the optimization COM-objects.

iAgent checks the iManager administrative database for hosts that have the automatic startup property enabled and starts them. As each host publishes its information to the database upon starting up, iAgent detects the host presence and activates the host's publishing mechanism. iAgent also creates and manages instances of **SysCounters** to retrieve information about every server on which a host resides. There will be one object per server. Simultaneously, iAgent creates a single instance of **MetiProc,** in order to calculate the statistics for each host.

iAgent can also be used to start other applications. Simply edit the text file `3rdPartyApp.ini`, located in the iManager installation folder. Use the following format for your entries:

[<Application Name>]

Active=<YES/NO>

Welcome

Welcome to MetiLinx iManager Developer 2.2 and MetiLinx™ technology—Making the Internet Powerful!™

MetiLinx digital technology tools make the Internet a more powerful place to do business, by delivering speed, flexibility, stability, dependability, and optimization to commercial, web-based systems.

Look for new things to come from MetiLinx as we continue to expand our line of optimization and development enhancement products. Please visit our Web site at www.metilinx.com.

Firewall Settings

Client application processes may cross a firewall to access an iManager host. When you create an iManager host, you must assign it a port number to enable access to the host through a firewall.

By default, the iManager host configuration process initially uses connection port 1024 for the first, established iManager host. Once connectivity is established and security access authorized, the host hands off the session to an available socket, then continues to listen for port 1024 session requests.

For each additional host you create with iManager, you must assign a different port number. These port assignments enable subsequent hosts to receive session requests.

Note:

Set up port 1024 and subsequent ports to use the same rules applied to the HTTP connection port 80 on your firewall. Use ports 1025 through 5000 with the exceptions of ports 3012 and 4012. iManager reserves these ports for the TLO and SLO components.

Accessing Database Servers through Business Rule Objects

iManager provides the COM object, QueryHost (defined in QueryObject.DLL), to grant Business Rule Objects (BRO) access to the database servers using the same connection assigned to the client making the request. The BRO must create an instance of QueryHost and then pass the SQL statement to the created instance, in order to retrieve the data.

For this purpose, the QueryHost interface (**IQueryHost**) exports a method named RequestQuery. The prototype for this method is:

```
procedure RequestQuery(const CmdStr: WideString; CmdInfo: OleVariant;
                       var CmdResult: OleVariant); safecall;
```

where

CmdStr contains the SQL-statement to be executed.

CmdInfo is a parameter passed to the BRO by the host. It is a Variant array with six elements, namely:

- CmdInfo[0] = Name of the Server for QueryHost (as a wide string).
- CmdInfo[1] = used by iManager.
- CmdInfo[2] = reserved for future use.
- CmdInfo[3] = reserved for future use.
- CmdInfo[4] = reserved for future use.
- CmdInfo[5] = reserved for future use.

CmdResult returns the resulting data.

Accessing Database Servers through Business Rule Objects

iManager provides the COM object, QueryHost (defined in QueryObject.DLL), to grant Business Rule Objects (BRO) access to the database servers using the same connection assigned to the client making the request. The BRO must create an instance of QueryHost and then pass the SQL statement to the created instance, in order to retrieve the data.

For this purpose, the QueryHost interface (**IQueryHost**) exports a method named RequestQuery. The prototype for this method is:

```
procedure RequestQuery(const CmdStr: WideString; CmdInfo: OleVariant;
                       var CmdResult: OleVariant); safecall;
```

where

CmdStr contains the SQL-statement to be executed.

CmdInfo is a parameter passed to the BRO by the host. It is a Variant array with six elements, namely:

- CmdInfo[0] = Name of the Server for QueryHost (as a wide string).
- CmdInfo[1] = used by iManager.
- CmdInfo[2] = reserved for future use.
- CmdInfo[3] = reserved for future use.
- CmdInfo[4] = reserved for future use.
- CmdInfo[5] = reserved for future use.

CmdResult returns the resulting data.

Add Users/Groups from External Source

Administrators can import user IDs and users groups from several types of external sources. The success of the import procedure, however, depends on including the exact information required, as indicated below.

Source	The location from which the collection of user and user group information is taken. Enter the following source information based on the Source Kind: SQL Server — server name WinNT — NT domain name Oracle — service name LDAP — server IP address
Source Kind	A code representation of the source program. Valid source kinds include: WinNT (Windows NT Domain) SQL Server and Oracle (Database Server) LDAP (Server). When the source kind is selected, the iManager automatically generates a new source name identifier. This identifier is modifiable.
Source Name	A unique identifier for a source. The same as the Source ID that is part of a user's authentication credentials. Select New to create a new source name. iManager verifies that the name is unique. Do not check New if you want to use an existing source name.
Administrator ID	The user ID that has administrative authority to access the source.
Password	The password for the administrator ID..
Database/Directory Name	If necessary, depending on the kind of external source, specify a database name (for database servers) or a directory to from which to retrieve users and user groups information.
Provider	Select the corresponding database access provider or type a connection string.

Tip:

Specify only the corresponding source information indicated below:

Windows NT	Source: Domain name or IP address Source Kind: WinNT Source Name: A unique identifier
LDAP	Source: IP address of the LDAP server Source Kind: LDAP Source Name: A unique identifier
SQL Server	Source: IP address of the SQL server Source Kind: SQL Server Source Name: A unique identifier Administrator ID: Sysadmin (sa) or an equivalent administrator login. Verify type

of SQL server security.

Database: Name of the database from which the users and user groups are being taken.

Provider: SQLOLEDB.1, the current Microsoft OLE DB provider for SQL Server. Otherwise, use MSDASQL.1, the Microsoft OLE DB provider for ODBC

Oracle

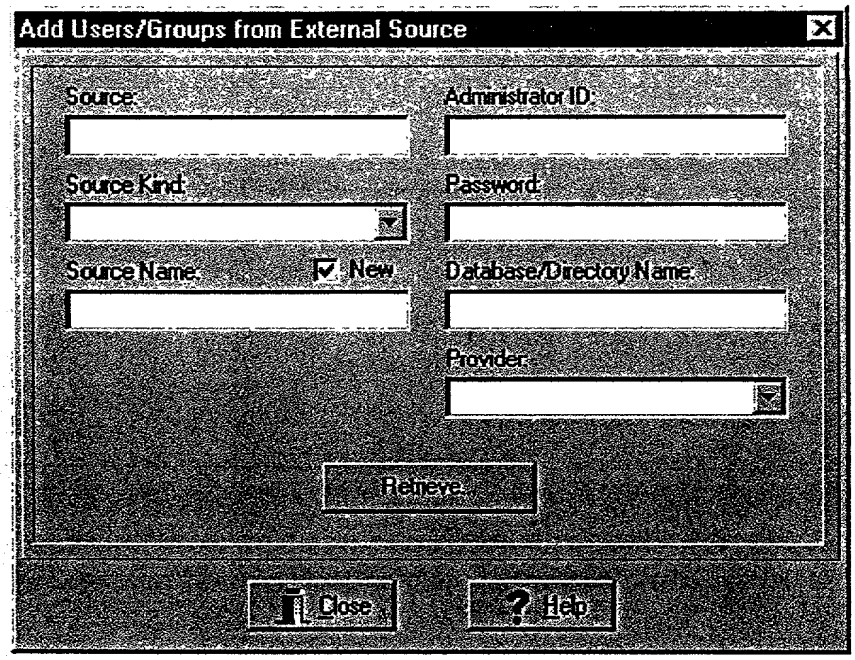
Source: The Oracle service name (Servername)

Source Kind: Oracle

Source Name: A unique identifier

Administrator ID: Use Internal or an equivalent administrator login.

Provider: MSDAORA.1 for the Microsoft OLE DB provider for Oracle

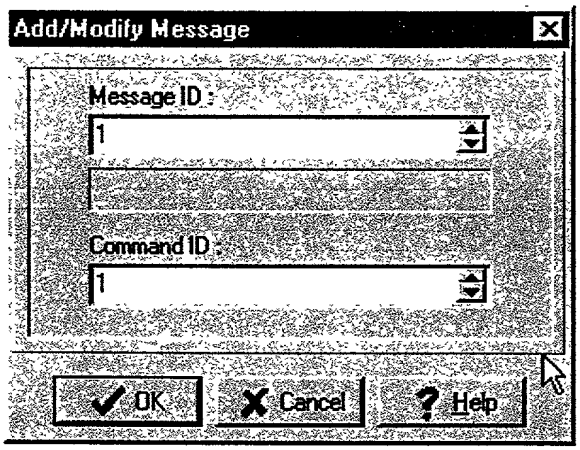


Additional Topics:

Microsoft OLE DB providers

Add/Modify Message

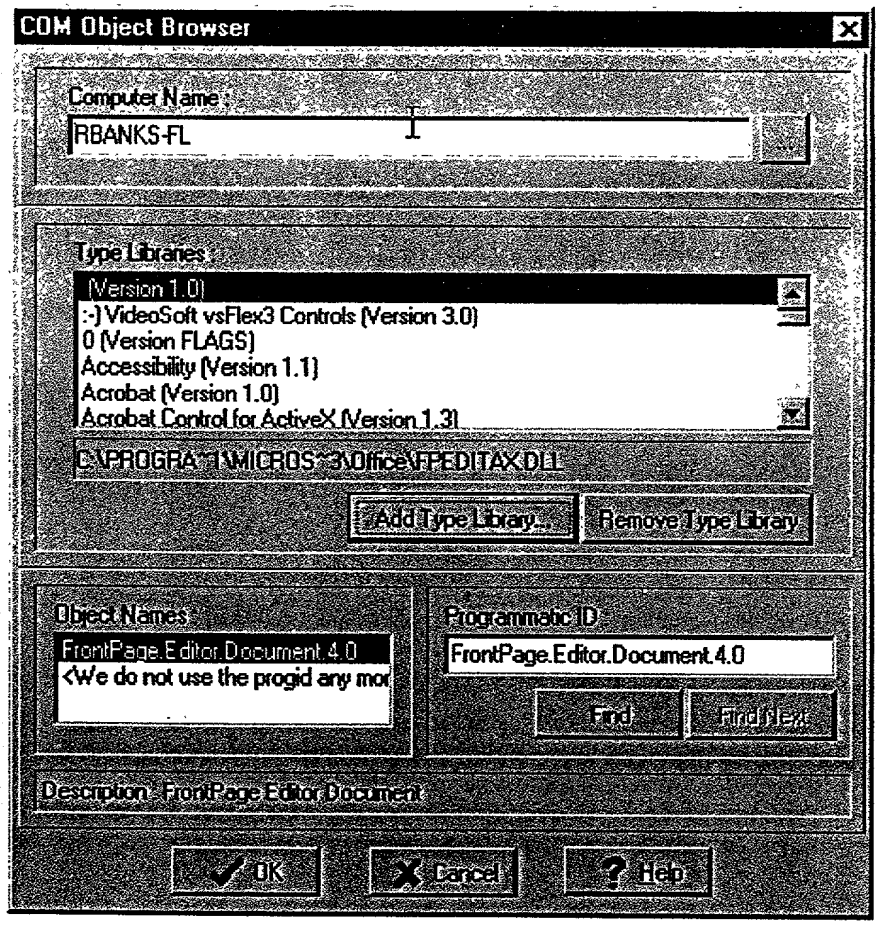
Message ID	Number (**DWORD**) associated to an alert definition.
Command ID	Unique identifier (**DWORD**) of a command.



Adding a COM Object to the Object Repository

1. Select **COM Objects** in **Object Repository> Business Rules Objects**.
2. Click **Action | Add Object**.

iManager opens the **COM Object Browser** dialog box.

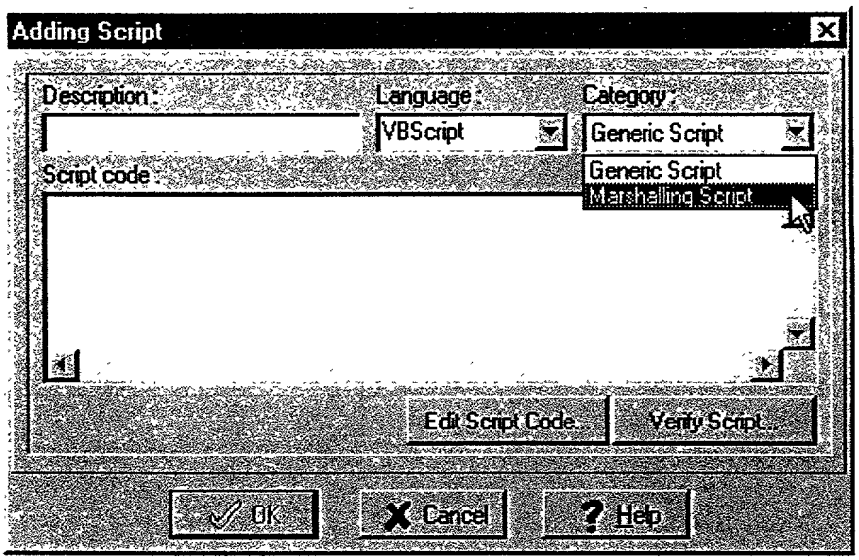


3. Select an item from the **Type Libraries** box.
4. Click **Browse** to find **Type Libraries** from another computer.
5. Click **Add Type Library**,
6. Select a type library you want to remove and click **Remove Type Library**.
7. Select an object from **Object Names**.
8. If the Type Library selected has numerous objects, you can easily search for a general object name by typing part of the object name in the Programmatic ID field and clicking **Find** and then **Find Next** until object is found.

Adding a Script to the Object Repository

1. Click **Scripts** located in **Object Repository > Business Rules Objects**.
2. Click **Action | Add Script**.

iManager opens the **Adding Script** dialog box.



3. Type script name in the **Description** box.
4. Select the language of the script.
5. Select the script type in the **Category** box (Generic or Marshalling).
6. Type the script code directly in the box, or click **Load Script Code** button to download a selected script.
7. Click **Verify Script** to test your script for errors.
8. Click **OK**.

Additional Topics

Learn about the Script Editor

Adding Connections to a Host

iManager allows the administrator to add or assign connection objects that are listed in the Object Repository to the iManager system hosts. This process consists of two-steps:

1. Adding a connection object to the Object Repository (if one that you want to use does not already exist), and then
2. Adding or assigning the connection to a host.

This topic focuses on Step 2.

To add a connection to a host (easy drag-and-drop operation)

1. Expand the **Hosts** folder, and then expand the folder of the host you wish to add a connection to.
2. Click **Connections** and drag the connection object in the right pane to the **Connections** folder of the host.

To add multiple connections to a host

1. Expand the **Hosts** folder, and then expand the folder of the host you wish to add a connection to.
2. Right-click **Connections**, and then click **Add Connection**.

   –or–

   Click **Connections**, and then on the **Action** menu click **Add Connection**.
3. In the **Add Connections to Host** dialog box, select the desired connection **Object Name** check boxes or click **Select All**
4. Click **Import selected** to add the selected connections to the host.

Additional Topics:

Adding Connection Objects

Adding Users to a Host

To grant a user ID access to a user host connection object, the user ID must already exist in the Host Users folder.

To add a host user (easy drag-and-drop operation)

1. Expand the **Hosts** folder, and then expand the folder of the host you wish to add a user to.
2. Click **Users** and drag the user object in the right pane to the **Host Users** folder of the host.

To add multiple host users

1. Expand the **Hosts** folder, and then expand the folder of the host you wish to add a connection to.
2. Right-click **Host Users**, and then click **Add Users to Host**.

   –or–

   Click **Host Users**, and then on the **Action** menu click **Add Users to Host**.
3. In the **Select users for the host** dialog box, select the desired host **User ID** check boxes or click **Select All**
4. Click **Import selected** to add the selected user IDs to the host.

Additional Topics:

Adding Users to iManager

Security

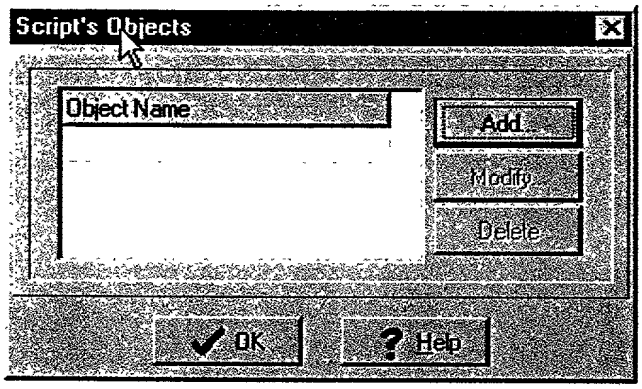
Adding Script Objects

Marshalling Scripts use script objects. In order to add a script object to a marshalling script, you must first add the object to the Object Repository. See Adding a Script to the Object Repository.

To add Objects to a Marshalling Script

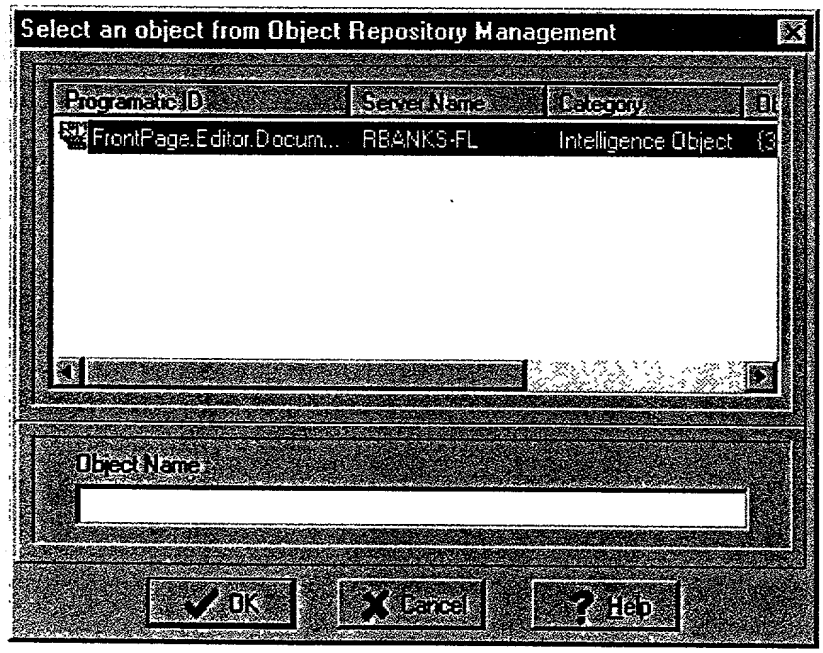
1. Select the script in **Scripts**, located under **Business Rules Objects**.

iManager opens the **Script's Objects** dialog box.



2. Click **Add**.

iManager opens the **Select an object from the Object Repository Management** dialog box.



3. Select the object you want to add to the script and enter a metaname in the **Object Name** box. The **Object Name** is the name used in the script code referring to this object's instance.
4. Click **OK**.

Repeat the above steps for each additional object.

Answer Messages

Message ID	Message Number	Description
imDone	$00000000	Acknowledge message from the host meaning operation successfully completed.
imError	$20000000	Included in the reply message ID whenever errors are raised.
imOverflow	$01010000	The information requested in a one-shot message did not fit into the message buffer.
imNoData	$01020000	A data request returned no data.
imDBError	$01040000	Error when trying to access a table or query.
ImSQLError	$01100000	The database provider returned an SQL-Error when trying to execute a query. Usually caused by syntax errors in the SQL statement.
imUnknownReport	$01200000	Trying to execute a dataset related request without opening a dataset first.
imDataNotFound	$02020000	imFind or imFindNext returned an empty page.
imDatasetOpen	$02040000	Trying to open a new dataset before closing the current one.
imCommError	$02080000	General communication error. Communication to the Host has been lost.
ImParamError	$02200000	Some of the parameters sent in the request are erroneous. (Example: a wrong page number for imGotoPage would generate this error message.)
ImFileError	$02400000	Requested file does not exist.
imFieldNotFound	$02800000	Wrong fieldname for imFind or imFindNext.
imTransactionError	$04010000	Transaction rules were violated.
imUnknownFile	$04080000	Requested file not found.
imLoginError	$04200000	Login/password error at login time.
ImReconnectionError	$04400000	Unable to reconnect to the requested host.
imUnknownMessage	$40000000	Message ID unknown.
imUnknownError	$01111111	Error of unknown cause.
ImLoginSourceError	$04800000	Error connecting to an authentication source.
imDBAccessError	$08000000	Unable to assign a dB connection for the user at login time.
imAdminDBError	$04100000	Unable to connect to the admin database (i.e. when login)

Client Objects

MetiLinx technology provides two different objects to be used by client applications to establish connections to hosts: ioRemote and ioCOMRemote. Each of these objects has an API (application programming interface) defined to allow clients to interact with iManager hosts and transmit data to and from the database servers.

ioRemote

ioRemote is a dynamic link library (ioRemote.DLL) that implements a set of functions to open a connection to a host, send/receive data to/from the host, close the connection and perform miscellaneous tasks (like checking connection status). All functions can be called using the **stdcall** calling convention, so it can be used from applications written in Delphi, C++, Visual Basic or Java.

Along with functions, iManager Messages must be used to retrieve and update data on your n-tier application platform. These messages have been designed to allow the most flexibility, scalability and functionality. iManager Messages are further detailed in this section.

ioCOMRemote

This object is actually an active DLL containing the definition and implementation of **MetiLinxClient**, a COM-object implementing the client functionality. MetiLinxClient COM-interface is implemented using the iManager messaging system rather than DCOM, so it enjoys the same speed and reliability as ioRemote. It is the ideal object for establishing a painless connection between web-servers or web applications, and a iManager host.

Client Software Requirements

- Windows NT Workstation 4.0
- Windows 98
- Windows 95
  1. Windows Socket 2 (Winsock)
  2. Dial-Up Networking 1.3 Performance and Security Upgrade Patch

Closing Connections to a Host

procedure Disconnect ; safecall

Description: Closes an existing connection to a host.

Closing Connections to Host

prodecure CloseConnection ;stdcall; export;

Description: Closes an existing connection to host.

Returns: No value.

Example in PASCAL:

```
Try
ClientID := LoginEx (pHostName, ServerAddr , pUsername, pPassword, LError, pPort);
...
Finally
    CloseConnection;
End;
```

The above example connects to the Host with the function LoginEx. After the job is done, it closes the connection by calling the procedure CloseConnection.

Code Sample

The example included here shows how to use `ioCOMRemote.dll` for executing your remote COM objects and for querying the DLO object at the server. The sample also shows how to connect to a host and invoke the DLO methods using a host message.

Note:

You need to import all of your COM objects using the IMetiLinx interface. If you wish to obtain a list of data servers for a given host, import the DLO object into iManager.

```
Dim Obj As Object

Private Sub Command1_Click()

 Caption = "Connecting..."
 Set Obj = CreateObject("ioComRemote.MetilinxClient")
 Call Obj.Connect("HOST1", "192.168.2.84", 1024, "username1", "password", "metilinx", status)
 Caption = "Connected to 192.168.2.84"
End Sub

Private Sub Command2_Click()
 Dim msgid   As Variant
 Dim res     As Variant
 Dim res1    As Variant
 Dim command As Variant
 Dim status  As Variant

 'executing a COM object that resides in the server side
 msgid = 2147483650# 'the first message after the last iManager reserved message
                     'corresponding to hex number 80000002
 command = "select * from authors"
 Call obj.Execute(msgid, command, res, status)
 Label1.Caption = "value >>>> " + res

 msgid = 2147483651# 'the second message after the last iManager reserved message
                     'corresponding to hex number 80000003
 command = "HOST1;" + "2400380870"
 Call obj.Execute(msgid, command, res, status) 'to obtain the data servers
 Label1.Caption = "DataServer 1>>>> " + res(0, 0) + " STATISTIC=" + res(0, 1)
 Call obj.Execute(msgid + 1, command, res1, status) 'to obtain the connection strings
 Label2.Caption = "ConnStr 1>>>> ID=" + res1(0, 0) + " CONNSTR=" + res1(0, 1)
 Set res = Nothing
 Set res1 = Nothing
 End Sub

Private Sub Command3_Click()
 Set obj = Nothing
 Unload Me
End Sub
```

```
Private Sub Form_Load()
  Caption = "Client using ioCOMRemote.DLL"
End Sub
```

Additional Topics:

Data Load Object

ioCOMRemote

COM (OLE/Automation) Errors

Start at 0x80000001 (-2147483647). For a complete list of error codes, refer to Microsoft documentation.

Additional Topics:

iManager errors

Configuring iManager: Security Information

Use an existing administrative user login to administrate iManager or create a new user.

1. Select the **Create New User** check box to create a separate administrative account to administrate the iManager database. Be sure to enter the Source ID (SID) previously indicated.
2. Enter a **Username** and **Password**, confirm the password, and then click **Next.**
3. Click **Run All** to create the database.

Configuring iManager: Create Connection Object

Using SQL 7

If you are using SQL Server 7 as your Database Management System (DBMS), then follow these instructions to create the iManager administrative database.

To create the administrative database

1. At the **iManager Developer 2.2 Configuration** window, select **Microsoft SQL Server**, then click **Next**.
2. Click **Edit Connection String** to create the ADO connection string.
3. Click the **Provider** tab, and select **Microsoft OLE DB Provider for SQL Server**.
4. Click **Next**.
5. Click the **Connection** tab, and select or enter a name in the **Server Name** box.
6. Enter a **Username** (typically, **SA**) and **Password**. (Verify the type of security SQL Server uses—Windows NT Only (Integrated) or SQL Server and NT.)
7. Click **OK** then click **OK**, again, at the **Create Connection Object** window.

Note:

Do not complete Step 3 of the Connection tab. If you do, you will receive error messages because the administrative database does not yet exist.

8. Click **OK** to test the connection object.
9. Enter the **Username** and **Password** you previously indicated.
10. Select the **Create New User** check box to create a separate administrative account to administrate the iManager database.
11. Enter a **Username** and **Password**, confirm the password, and then click **Next**.
12. Enter a **Name**, for example, MetiLinx, for the new iManager administrative database.
13. Click **Run All** to create the database.

Once the installation process builds the administrative database, it is complete.

Note:

Do not edit the connection string. The only time the connection string is altered, is when you reinstall the application. When you do, a new database overwrites the previous administrative database.

Using Oracle 8i

If you are using Oracle *8i* as your DBMS, then follow these instructions to create the iManager administrative database.

Configuring iManager to use an Oracle database requires running the iManager Oracle Database Constructor (`Mkoradb.exe`) at the database server to simplify the database creation process. The batch file completes the following processes on the Oracle database server:

- Creates and starts an Oracle instance
- Creates a database associated with the instance
- Adds the database to the listener file so it can accept client connections
- Configures the client connection

To Create the Oracle Database Using Mkoradb.exe

1. Open a command prompt at the database server.
2. Type **c:** or the drive letter where you installed iManager.
3. Type **cd program files\metilinx\metilinx enterprise 2.2** or the directory path where you installed IManager.
4. Type **mkoradb DBID DBNAME *password*** to run the batch file where the parameters represent the following usage:
   - **DBID** A four-character database system identifier (SID) (4 character limit)

- **DBNAME** The name of the database being created (8 character limit)
- ***password*** The password for the Internal (or the first user) of the database. This user is granted super user rights

5. To test the connection, type **vaw internal/*password*@dbname** at the command prompt.

Once the database constructor builds the administrative database and you have tested the connection, you may continue with configuring iManager.

Tip:

Use MTLX as the system (source) identifier and METILINX as the database name.

Configuring iManager to Use the Oracle Administrative Database

1. Return to the iManager installation on the host system.
2. Select **Oracle *8i*** as the administrative database.
3. Enter the Oracle Connection information:
   - **Server Name** Name of the new Oracle database
   - **SID** Source or system ID (use MTLX)
   - **Host Name** Name of the host on which the database (server) resides
4. At the **Create Connection Object** window, click **Edit Connection String** to create the ADO connection string.
5. At the **Provider** tab of the **Data Link Properties** window, select **Microsoft OLE DB Provider for Oracle**.
6. Click **Next**.
7. At the **Connection** tab, select or enter a name in the **Server Name** (database name) box.
8. Enter information to logon to the database: **Username** (use Internal) and **Password**.
9. Click **OK** to test the Connection Object.
10. Enter the same **Username** and **Password** to logon to the database. If the test is successful, the **MetiLinx Enterprise 2.2 Configuration** window appears. If the test is unsuccessful, verify the Server Name, SID, and Host Name.
11. Select the **Create New User** check box to create a separate administrative account to administrate the iManager database. Be sure to enter the Source ID (SID) previously indicated.
12. Enter a **Username** and **Password**, confirm the password, and then click **Next.**
13. Click **Run All** to create the database.

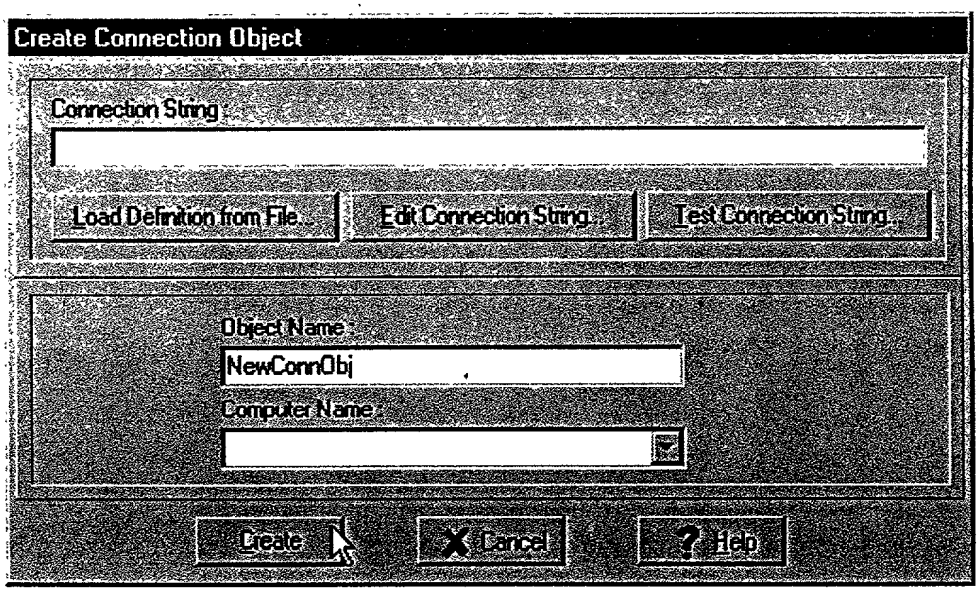
Once the installation process configures the iManager administrative database, it is complete.

Note:

Do not edit the connection string. The only time the connection string is altered, is when you reinstall the application. When you do, a new database overwrites the previous administrative database.

Create Connection Object

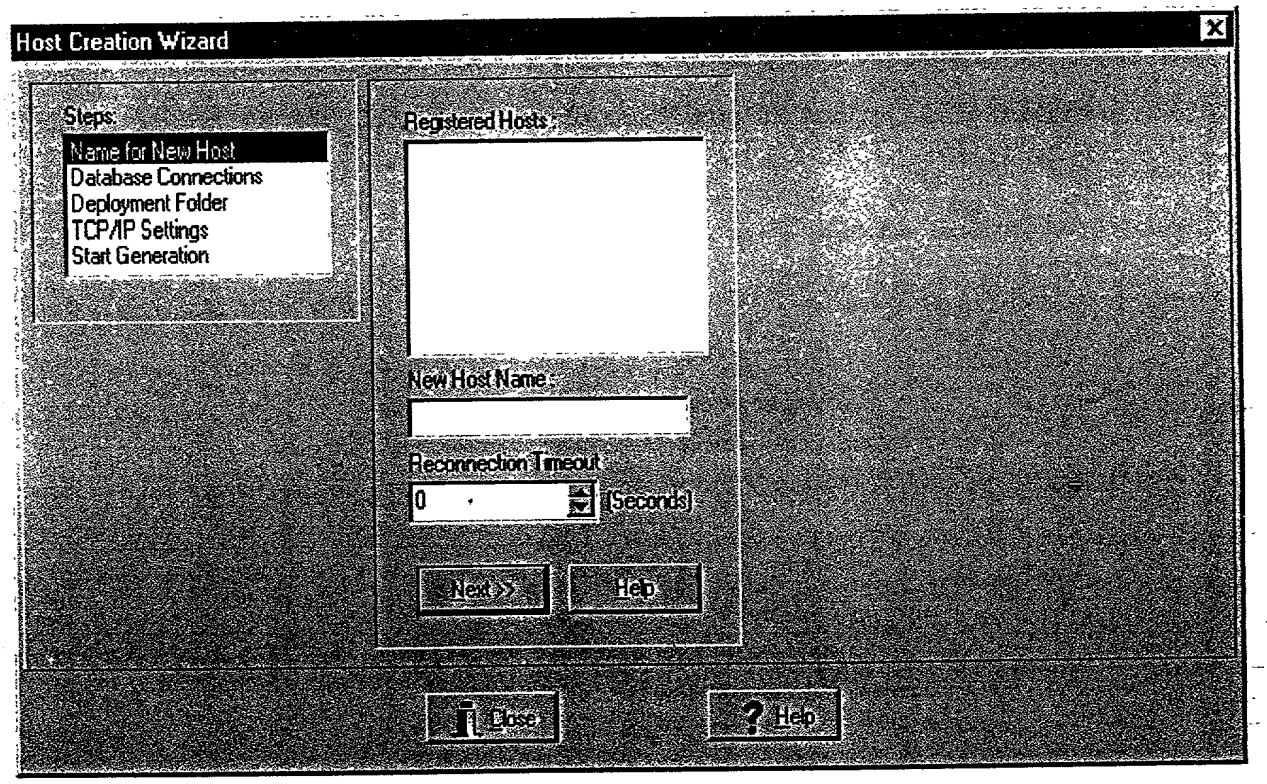
Connection String	The path specification to OLE DB or ODBC data sources.
Load Definition from File	Select if you are specifying a Microsoft Data Link connection, you must load an existing data link (.udl) file
Edit Connection String	Select to specify information about source and destination OLE DB data sources. The information includes server names, format and location of the data, and passwords. The connection is established by the first task that uses the connection. A data source connection can specify information about an ODBC data source when using the Microsoft providers.
Test Connection String	Select to test connectivity to the data source.
Object Name	The name of the connection string object.
Computer Name	Select the computer on which the data source resides.



Creating a Host

The Host Creation Wizard guides you through the five-step process of creating an iManager host. These process steps are:

1. Naming the host
2. Creating the database object
3. Creating the deployment folder
4. Establishing the TCP/IP settings
5. Generating the host



The Host Creation Wizard

To access the Host Creation Wizard

- On the **MetiLinx iManager** window, right-click **Hosts**, and then choose **Add New Host**.

  –or–

  On the **Action** menu, click **Add New Host**.

To name the host

1. Enter the **New Host Name** (20 character–alpha-numeric– maximum, first character must be a letter).
2. In the **Reconnection Timeout** box, type or select a number.
3. Click **Next**.

To create the database connection object

1. Click **Create Connection Object**.
2. On the **Create Connection Object** dialog box, type an **Object Name**, and then click **Edit Connection String**.
3. You **do not** need to type or select a **Computer Name**.

4. On the **Data Link Properties** dialog box, click the **Provider** tab.
5. Select the appropriate **OLE DB Provider**, and then click **Next**.
6. If you are setting up a connection to an Oracle database, skip **Step 7**.
7. On the **Connection** tab, type or select a **Server Name**.
8. You may click **Refresh** to update the list of available servers.
9. Enter a **Username** and **Password** to long on to the server. (Verify the type of security SQL Server uses—Windows NT Only (Integrated) or SQL Server and NT.)
10. Type or select the database you are establishing a connection to.
11. Click **OK** then click **OK**, again, at the **Create Connection Object** window.
12. At the **Connection Object Parameters** dialog box, verify the **Provider** and choose the **Security** parameter you would like.
13. After verifying the connection object parameters on the **Create Connection Object** dialog box, click **Create** to create and test the connection object.
14. Enter a **User Name** and **Password**.

Return to the **Host Creation Wizard** to create additional database connection objects.

To add the database connection object

1. Select from the list of **Available Connection Objects** the connection object(s) you wish to add to the **Selected Connection Objects**.
2. Click **Add** or **Add All**.
3. Click **Next**.

To create the deployment folder

1. Verify the **Drive** where you installed iManager.
2. Accept the default file location, `C:\Program Files\MetiLinx\MetiLinx Enterprise 2.2\Hosts`, and then click **Next**.

You may create an alternative folder location. MetiLinx recommends that you accept the default location.

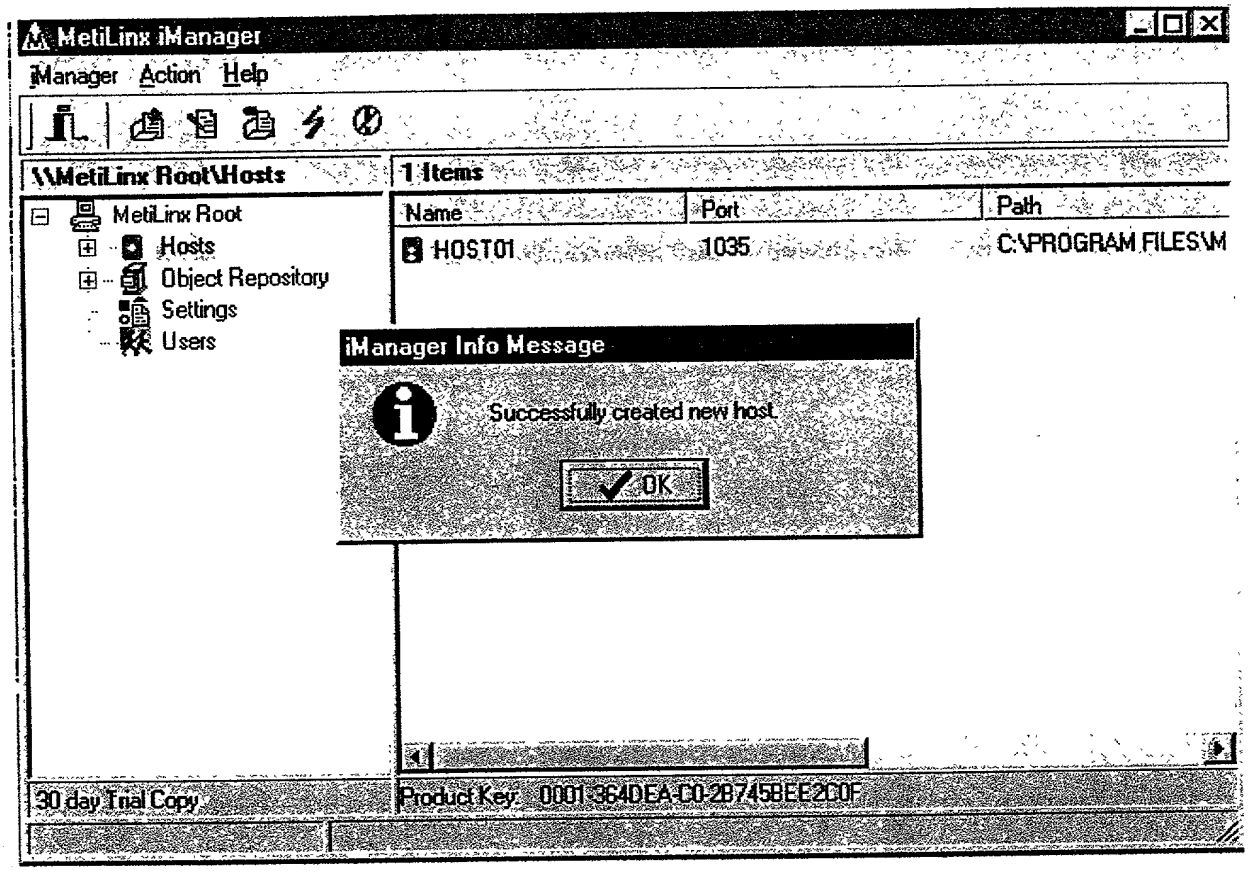
To establish the TCP/IP settings

1. Type or select a TCP/IP Port.
2. Select the IP address of the iManager application server.
3. All static IP addresses for the server are listed.
4. Click **Next** to proceed to the final step in the process.

To generate the host

1. Verify the Host Creation Options, and then click **Finish**.
2. To change the options, click **Previous** and proceed with the above steps.

An iManager Info Message will indicate the successful creation of the host.



Note:

To enable host logging, you must modify the host.

Additional Topics:

Modifying a host

Log Settings

Data Load Object (DLO)

DLO is a COM object contained in the `Dlo.exe` file supplied with version 2.2, which requires you to develop your own COM interface. This object provides information about data servers and connections associated with a host. It supports user's business rules implementation by enabling more dynamic opening of database connections.

By creating your own interface, you can use the DLO to replicate objects to databases and recover failed database connections. The DLO enhances your use of the iManager optimization mechanism by providing a sorted list of connection strings based on server-load statistics. It implements the IDataLoadObject and IMetiLinx interfaces.

Data Messages

Message ID	**imGetSQLData**
Message Number	$00000210
Explanation	This message performs the query requested like SELECT and keeps an open dataset for further requests. It also sends the first page of the data result set.
Required Payload	Number of lines per page and the query statement to be performed. Both these parameters should be separated by CRLF.
Payload Return	The first page of the data result set with each field separated by TAB and each line or record separated by CRLF. Records per page might be adjusted to avoid page overflow.
Code Example (PASCAL)	BufferStr := '20' + #13#10 + 'SELECT * FROM UserInfo' + #13#10 StrPCopy(Buffer , BufferStr); MsgID:= imGetSQLData SendMsg(MsgID, ClientID, Buffer);
Result	The above example returns the first page with 20 lines or records of the information in the UserInfo table and keeps an open dataset with all the pages of the query result.
Function Word	SendMsg

Message ID	**imFirstPage**
Message Number	$00000220
Explanation	This message is used to get the first page of the SQL query result dataset. This message can be used only after the message imGetSQLData is sent which keeps track of all the pages of the result set.
Required Payload	No value.
Payload Return	First page of the query result dataset with each field separated by TAB and each record separated by CRLF.
Code Example (PASCAL)	StrCopy(Buffer,#0); MsgID:= imFirstPage SendMsg (MsgID, ClientID, Buffer);
Result	The above example gets the first page of the data result set of the performed query.
Function Word	SendMsg

Documentation Key for Code Samples

ClientID:	The unique identifier for the client, which is returned by the login function.
HostName:	Null terminated string containing the Name of the Host as defined in iManager (in all caps).
LoginError:	Null terminated string that will be returned in case of error. Space for resulting string should be reserved by the client application.
Msg:	Null terminated string containing the message data sent or received.
MsgID	The message identifier sent when an action is requested from the Host. After executing the action, the host will return a different MsgID indicating the resulting status.
OLEResult:	Variant containing retrieved data.
Password:	Null terminated string containing the client's Password. Part of the user credentials.
Port:	Port number used to establish the initial connection between Host and client.
ServerAddr:	Null terminated string containing the IP address of the Host or the Name of the Server.
Size:	Size of data returned in the "Msg" parameter.
SourceID:	Null terminated string containing the client's SourceID. Part of the user credentials. NIL (NULL pointer) means **DEFAULT** (proprietary source).
UserName:	Null terminated string containing the client's User Name. Part of the user credentials.

Deleting a Connection Object

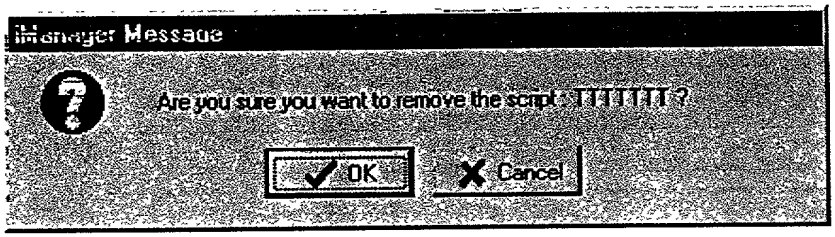
1. At the **MetiLinx** tree, click **Object Repository**; then click **Connection Objects**.
2. In the right pane, right-click the **Connection Object** you wish to delete; then click **Delete Connection Object**.
3. In the **iManager Message** dialog box, click OK to confirm the deletion.

Deleting a Script

Delete a script from the Object Repository if it is no longer in use.

1. Click **Scripts** located in **Object Repository > Business Rules Objects**.
2. Select the script you want to delete.
3. Click **Action | Delete Script**.

The **iManager Message** confirmation below appears.



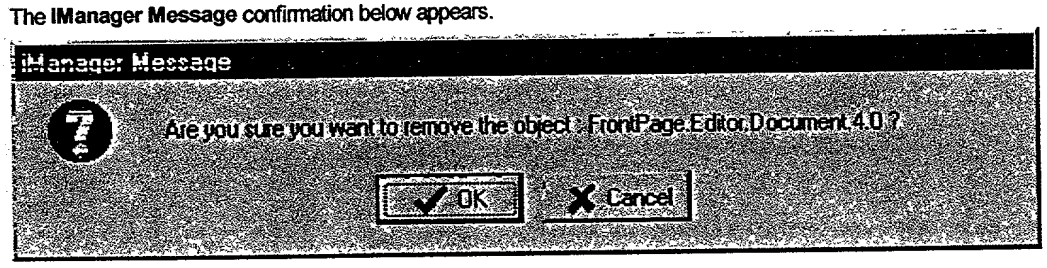
4. At the iManager message, click **OK** to complete the deletion.

Deleting an Object

Delete an object from the Object Repository if it is no longer in use.

1. Click **COM Objects** located in **Object Repository > Business Rules Objects**.
2. Click the object you want to delete.
3. Click **Action | Delete Object.**

The **IManager Message** confirmation below appears.



4. Click **OK** in the iManager message dialog box, confirm the object deletion.

Contents of the Deployment Folder

Each Host deployment folder contains the subdirectories and files described in the table below.

Directory	File(s)	Description
BIN	iEvent.exe	This executable applies the events and changes to the Host settings.
	ih<HOSTNAME>.bin	These are the configuration files for your Host system as set in iManager and are the supporting executable files for this Host.
UPDATE	Empty	This directory is where downloadable files for client applications are placed. The imGetFile message is used to accomplish this through your client application. This message and its use are explained later in the Messages section of this document.

Note

The contents in the BIN folder should not be moved or modified under any circumstances.

To Install MetiLinx iManager Developer 2.2

1. Close all programs, including virus-checking programs.
2. On the **Start** menu, select **Run**.
3. Click **Browse** to locate your Download folder.
4. Type or select the file name **metilinxenterprise2.2.exe**, and then click **Open** to run the file.
5. Follow the instructions of the InstallShield Wizard.
   - Choose Destination Folder
     By default, iManager 2.2 is installed in C:\Program Files\MetiLinx\MetiLinx Enterprise 2.2.
   - Select Program Folder
     By default, the program folder is **MetiLinx Enterprise 2.2**.

Note:

You must have full Administrator rights to the local machine.

You are now ready to create the iManager administrative database.

Additional Topics:

To create the iManager administrative database using SQL Server 7

To create the iManager administrative database using Oracle 8i

Data Load Object (DLO)

DLO is a COM object contained in the `Dlo.exe` file supplied with version 2.2, which requires you to develop your own COM interface. This object provides information about data servers and connections associated with a host. It supports user's business rules implementation by enabling more dynamic opening of database connections.

By creating your own interface, you can use the DLO to replicate objects to databases and recover failed database connections. The DLO enhances your use of the iManager optimization mechanism by providing a sorted list of connection strings based on server-load statistics. It implements the IDataLoadObject and IMetiLinx interfaces.

Data Load Object (DLO)

DLO is a COM object contained in the `Dlo.exe` file supplied with version 2.2, which requires you to develop your own COM interface. This object provides information about data servers and connections associated with a host. It supports user's business rules implementation by enabling more dynamic opening of database connections.

By creating your own interface, you can use the DLO to replicate objects to databases and recover failed database connections. The DLO enhances your use of the iManager optimization mechanism by providing a sorted list of connection strings based on server-load statistics. It implements the IDataLoadObject and IMetiLinx interfaces.

Documentation Key for Request Messages

Message ID	Message Name
Message Number	Message Number used in the LongInt portion of the message.
Explanation	Description of the use of the message.
Required Payload	The payload required in the message.
Payload Return	The payload or value returned by the Host, if any.
Code Example	An example of how this message would be used in coding language.
Result	Shows the result you would expect from the Code example above.
Function Word	Name of the function used to send this message ID.

What's New with MetiLinx iManager Developer 2.2

MetiLinx Enterprise 2.1 has branched out to MetiLinx iManager Developer 2.2 (iManager) and MetiLinx iSystem Enterprise 2.2 (iSystem). iManager is a development tool that aids developers in building e-business systems. These systems subsequently inherit the MetiLinx optimization technology. This release of iManager delivers a broad range of object support and management in an open environment.

✓ Dual interface support for scripting languages

MetiLinx iManager Developer 2.2 supports the *IDispatch* interface to enable developers working in scripting languages, such as VBScript and JScript, to access the iManager Repository components.

✓ New! Script Editor

The new Script Editor offers basic editing support, enabling developers to edit code from generic and marshalling scripts.

✓ Enhanced remote COM management

With MetiLinx iManager Developer 2.2, MetiLinx continues to build and expand upon the remote COM management functionality and open environment of MetiLinx Enterprise 2.1. Enhanced features include:

- Improved object streaming
  - Enables remote object handling without the need to register the client
  - Promotes code efficiency
  - Supports local and remote calling on client
- Object repository management
- Access to host-level information, such as, connection objects, repository objects, and user connections.

Easy Steps to iManager Implementation

1. Create Hosts with the Create Hosts Wizard
2. Create Connection Objects in the Object Repository
3. Create Global Users
4. Add Users to Hosts
5. Add Connections to Hosts
6. Add User Access to Connections
7. Implement Business Rules
   - Add COM Objects to the Object Repository
   - Add Scripts to the Object Repository
8. Establish Universal Settings

Additional Topics:

Installing iManager

COM (OLE/Automation) Errors

Start at 0x80000001 (-2147483647). For a complete list of error codes, refer to Microsoft documentation.

Additional Topics:

iManager errors

Establishing Log Settings

By default, the host log is not activated during the creation of a host. Administrators, therefore, must manually activate logging, and then configure the Log Viewer for each host to filter log the desired information.

To activate a host log

1. Click the **Host** folder to view the list of hosts.
2. Right-click the host you wish to active logging on, and then click **Modify Host**.
3. Click Log Settings and select the **Enable Log** check box.
4. Select the desired option settings.

To configure the log viewer

1. Click the **Host** folder to view the list of hosts.
2. Right-click the host you wish to active logging on, and then click **Modify Host**.
3. Click Log Viewer Settings and select the desired option settings.

Establishing Universal Log Settings

Use the Settings module to modify Agent Refresh Time, Log Refresh Time, Log View Entries, and Screen Refresh Time for Hosts.

1. Click **Settings**, then right-click the setting you want to modify.
2. Click Modify Settings.
3. Enter the parameter values you want to use.

Firewall Settings

Client application processes may cross a firewall to access an iManager host. When you create an iManager host, you must assign it a port number to enable access to the host through a firewall.

By default, the iManager host configuration process initially uses connection port 1024 for the first, established iManager host. Once connectivity is established and security access authorized, the host hands off the session to an available socket, then continues to listen for port 1024 session requests.

For each additional host you create with iManager, you must assign a different port number. These port assignments enable subsequent hosts to receive session requests.

Note:

Set up port 1024 and subsequent ports to use the same rules applied to the HTTP connection port 80 on your firewall. Use ports 1025 through 5000 with the exceptions of ports 3012 and 4012. iManager reserves these ports for the TLO and SLO components.

Hardware Requirements

- 300 MHz Pentium™ II processor
- 120 MB RAM or more
- 100 MB free disk space

Host Creation Wizard: Database Connections

To create the database connection object

1. Click **Create Connection Object**.
2. On the **Create Connection Object** dialog box, type an **Object Name**, and then click **Edit Connection String**.
3. You **do not** need to type or select a **Computer Name**.
4. On the **Data Link Properties** dialog box, click the **Provider** tab.
5. Select the appropriate **OLE DB Provider**, and then click **Next**.
6. If you are setting up a connection to an Oracle database, skip **Step 7**.
7. On the **Connection** tab, type or select a **Server Name**.
8. You may click **Refresh** to update the list of available servers.
9. Enter a **Username** and **Password** to long on to the server. (Verify the type of security SQL Server uses—Windows NT Only (Integrated) or SQL Server and NT.)
10. Type or select the database you are establishing a connection to.
11. Click **OK** then click **OK**, again, at the **Create Connection Object** window.
12. At the **Connection Object Parameters** dialog box, verify the **Provider** and choose the **Security** parameter you would like.
13. After verifying the connection object parameters on the **Create Connection Object** dialog box, click **Create** to create and test the connection object.
14. Enter a **User Name** and **Password**.

Return to the **Host Creation Wizard** to create additional database connection objects.

To add the database connection object

1. Select from the list of **Available Connection Objects** the connection object(s) you wish to add to the **Selected Connection Objects**.
2. Click **Add** or **Add All**.
3. Click **Next**.

Host Creation Wizard: Deployment Folder

To create the deployment folder

1. Verify the **Drive** where you installed iManager.
2. Accept the default file location, `C:\Program Files\MetiLinx\MetiLinx Enterprise 2.2\Hosts`, and then click **Next**.

You may create an alternative folder location. MetiLinx recommends that you accept the default location.

Contents of the host deployment folder

Each Host deployment folder contains the subdirectories and files described in the table below.

Directory	File(s)	Description
BIN	iEvent.exe	This executable applies the events and changes to the Host settings.
	ih<HOSTNAME>.bin	These are the configuration files for your Host system as set in iManager and are the supporting executable files for this Host.
UPDATE	Empty	This directory is where downloadable files for client applications are placed. The imGetFile message is used to accomplish this through your client application. This message and its use are explained later in the Messages section of this document.

Note

The contents in the BIN folder should not be moved or modified under any circumstances.

Host Creation Wizard: Name for New Host

To name the host

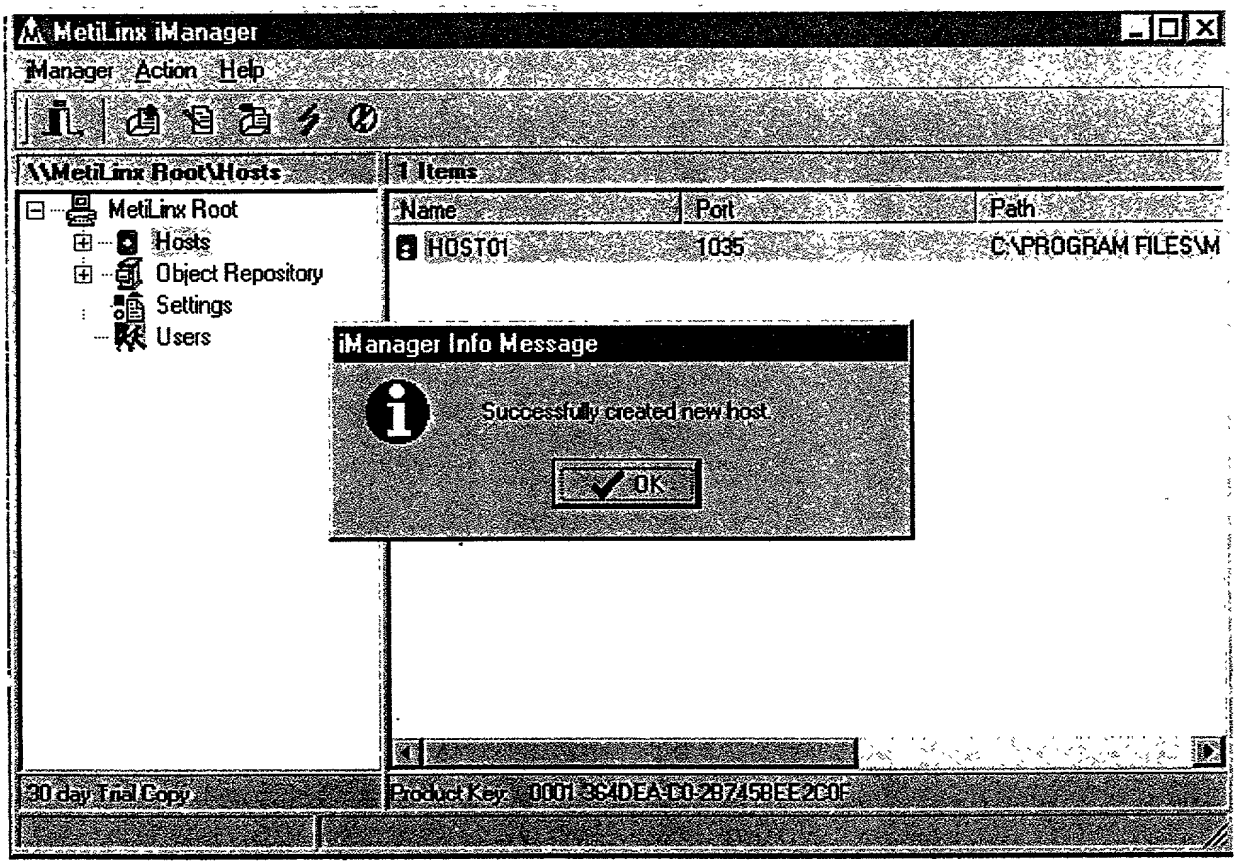
1. Enter the **New Host Name**.
2. In the **Reconnection Timeout** box, type or select a number.
3. Click **Next**.

Host Creation Wizard: Start Generation

To generate the host

1. Verify the **Host Creation Options**, and then click **Finish**.
2. To change the options, click **Previous** and proceed with the above steps.

An iManager Info Message will indicate the successful creation of the host.



Host Creation Wizard: TCP/IP Settings

To establish the TCP/IP settings

1. Type or select a TCP/IP Port.
2. Select the IP address of the iManager application server.
3. All static IP addresses for the server are listed.
4. Click **Next** to proceed to the final step in the process.

Host Objects

After successfully executing the Host Setup Wizard, a complete host system is created. Two objects are a part of this new system: **ioHost** and **iEvent**. Since there is no direct user interaction with these objects, no interface description is provided. Settings for both objects are modified through iManager and data is requested through remote connections.

ioHost

The ioHost Object instantly creates a complete host system on your server based on the parameters you select during the iManager Host creation process. The ioHost Object works with the ioRemote Object to create your custom application Object.

iEvent

The iEvent Object is in charge of logging host events and communications with iManager. This object intelligently gathers these events so it does not impede the performance between your client and Host applications.

The Object Repository

The  Object Repository is the module that provides COM object management. It includes functionality for registering, adding, removing and modifying objects. It also allows interface verification and instantiation, as a warranty of object availability.

iManager messages are pre-structured messages, functions and procedures that simplify the programming of your client application. They have been specifically designed for flexibility, scalability, and purpose. The built-in messaging protocol allows developers to extend iManager COM object capabilities with custom messages to interact with existing objects. iManager separates COM-Objects into two categories:  Connection Objects and  Business Rules Objects

Connection objects are solely used by the host system to establish connections to database servers using ODBC and OLE DB-connections.

Business Rules Objects refers to objects created outside of iManager and which will be accessed through the COM-metaphor. For these objects to be accessible from iManager, they must implement the IMetiLinx interface or use a marshalling script compliant with iManager specifications.

To use COM-objects, developers create new message IDs and associate them with actions and objects. These message IDs and associations are global for iManager and are stored in the  Object Repository. At the User's option, iManager can verify the integrity of object definitions by checking its presence and interface implementation.

There are three ways COM-objects can be used by a Host.

1. Direct access to COM-objects implementing IMetiLinx interface.
2. Access through a Marshalling Script to generic COM-objects kept in the repository.
3. Execution of a Generic Script kept in the  Object Repository.

Prototype of the interface IDataLoadObject

```
procedure getDataServers    ( const Host          : WideString;
                              ClientID            : OleVariant;
                              out DataServerList  : OleVariant); safecall;
```

```
procedure getConnStrings    ( const Host          : WideString;
                              ClientID            : OleVariant;
                              out ConnStrList     : OleVariant); safecall;
```

```
procedure closeADOconn      ( ConnID              : OleVariant); safecall;
```

```
procedure openADOconn       ( ConnID              : OleVariant;
                              out ConnADO         : OleVariant;
                              ConnOpenedID        : OleVariant); safecall;
```

ProcessCommand	(CmdID	: LongWord;	// = 1 (to pull the data servers list)
	const CmdStr	: WideString;	// = <Host name>;<ClientID> Example: CmdStr = 'Host1;1277608648'
	CmdInfo	: OleVariant;	
	var CmdResult	: OleVariant); safecall;	//A two dimensional array with the data servers information

Description: This additional object also implements the interface IMetilinx, which is accessible through the function.

ProcessCommand	(CmdID	: LongWord;	// = 2 (to pull the connection string list)
	const CmdStr	: WideString;	// = <Host name>;<ClientID>
	CmdInfo	: OleVariant;	//
	var CmdResult	: OleVariant); safecall;	// A two dimensional array with the data servers information

Description: This additional object also implements the interface IMetilinx, which is accessible through the function.

ProcessCommand	(CmdID	: LongWord;	// = 3 (to open an ADO connection using a connection ID)
	const CmdStr	: WideString;	// <ConnectionID>
	CmdInfo	: OleVariant;	
	var CmdResult	: OleVariant); safecall;	// A two dimensional array containing the ADO connection object and a connection-opened-ID.

Description: This additional object also implements the interface IMetilinx, which is accessible through the function.

ProcessCommand	(CmdID	: LongWord;	// = 4 (close a currently open connection)
	const CmdStr	: WideString;	// = <ConnectionOpenedID>
	CmdInfo	: OleVariant;	//
	var CmdResult	: OleVariant); safecall;	// = No values returned

Description: This additional object also implements the interface IMetilinx, which is accessible through the function.

Note:

Please observe the parameter differences when using different CmdID values.

Data Load Object (DLO)

DLO is a COM object contained in the `Dlo.exe` file supplied with version 2.2, which requires you to develop your own COM interface. This object provides information about data servers and connections associated with a host. It supports user's business rules implementation by enabling more dynamic opening of database connections.

By creating your own interface, you can use the DLO to replicate objects to databases and recover failed database connections. The DLO enhances your use of the iManager optimization mechanism by providing a sorted list of connection strings based on server-load statistics. It implements the IDataLoadObject and IMetiLinx interfaces.

Configuring iManager: Create Administrative Database

In this step, iManager builds the administrative database for Microsoft SQL 7.

1. In the **Database Name** field, enter a database name to create.
   Note: Do not check the boxes in the **Create Database** field. They will be completed automatically.
2. Click **Next**.
3. Click **Yes** when prompted to create database scripts.

The **InstallShield Wizard** will indicate that the iManager installation is complete. Before you can use the program, you must restart your computer.

iManager Errors

The Status parameter of the ioCOMRemote functions returns the following errors:

Code	Error
0	No error.
-20000	Unexpected error.
-20001	Error getting data from the Host.
-20002	Error writing to stream.
-20003	Error reading from stream.
-20004	Equivalent Hosts not found.
-20005	Host not found.
-20006	Error sending message.
-20007	Error in parameters.
-20008	Reconnection error.
-20009	Host name error.
-20010	Error accessing administrative database.
-20011	Database server error.
-20012	Username/Password error.
-20013	Error Connecting to Host.
-20014	Server name/Port incorrect.
-20015	Error Connecting to TLO.
-20016	Error looking for equivalent hosts.
-20017	Error in function Execute.
-20018	Internal error sending message.
-20019	Object not registered repository.
-20020	Object not in table.
-20021	Object not in remote table.
-20022	Interface not registered locally.
-30001	Error invoking object function.
-30002	Error getting memory for Dispatch Parameters structure.
-30003	Error freeing memory for Dispatch Parameters structure.
-30004	Error getting Parameter List.
-30005	Variant array has no dimensions.
-30006	Invalid Function Name.
-30007	Error setting the Parameter List.
-30008	Invalid variant type conversion getting parameters.
-30009	Invalid variant type conversion setting parameters.
-30010	Error creating IProvideClassInfo interface.
-30011	Error in IDispatch interface.
-30012	Error getting function description.
-30013	Error freeing function description.
-30014	Function description is empty.
-30015	Invalid Parameter List.
-30016	Invalid Interface Name.
-30017	Error trying to execute IDispatch.Invoke.
-30018	Error Saving IPersistStream.

-30019	Error getting ITypeInfo instance of the library.
-30020	Error streaming object.
-30021	Error saving variant to stream.
-30022	Error saving object properties.
-30023	Error loading variant from stream.
-30024	Error loading the variant from a IPersistStream.
-30025	Stream kind not allowed.
-30026	Error loading from stream.
-30027	Error saving stream to variant.
-30028	Error filling the Dispatch Parameters structure.
-30029	Error verifying the object in Repository.
-30030	Error getting the list of implemented interfaces from IDispatch.
-30031	Error getting the list of implemented interfaces from ClassInfo.
-30032	Error getting the ITypeLib interface.
-30033	Error in InvokeDotNotation function.
-30034	Error in SearchInterfaceName function.
-30035	Error in dot Notation.
-30036	Error getting ITypeLib interface of the object.
-30037	Error filling the Parameters list to pass it to invoke function.
-30038	Error filling the Named Argument list to pass it to invoke function.
-30039	Error in SaveCollectionToArray function.
-30040	Error in SaveArrayToCollection function.
-30041	Error in SaveDispatchObj function.
-30042	Error in Create function of TObjPerformer.
-30043	Error in GetInterfaceList function.
-30044	Error in GetMemberList function.
-30045	Error in GetParameterList function.
-30046	Error in SaveRecordSet function.
-30047	Error in LoadRecordSet function.

Additional Topics:

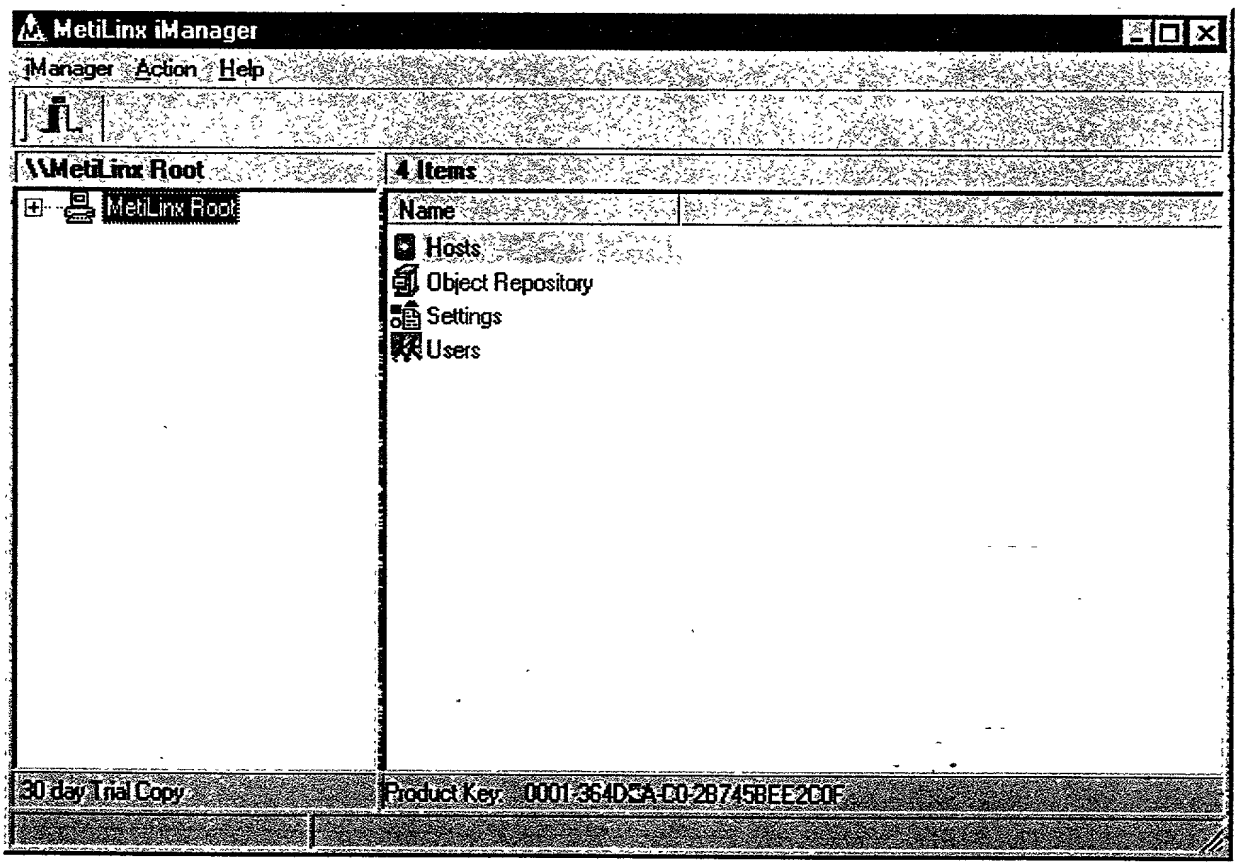
COM (OLE/Automation) Errors

iManager Toolbar Buttons

Button	Command
	Add
	Add New User/External Source
	Delete / Remove
	Exit / Close
	Filter
	Modify
	Run Host
	Stop Host
	Verify
	View All
	Object/Script Messages
	Script Objects

iManager Window

This is the main iManager window. Click on its various areas to learn more about its features.



Additional Topics:

Learn about the Object Repository

Prototype of the interface IDataLoadObject

```
procedure getDataServers    ( const Host          : WideString;
                              ClientID            : OleVariant;
                              out DataServerList  : OleVariant); safecall;
```

```
procedure getConnStrings    ( const Host          : WideString;
                              ClientID            : OleVariant;
                              out ConnStrList     : OleVariant); safecall;
```

```
procedure closeADOconn      ( ConnID              : OleVariant); safecall;
```

```
procedure openADOconn       ( ConnID              : OleVariant;
                              out ConnADO         : OleVariant;
                              ConnOpenedID        : OleVariant); safecall;
```

```
ProcessCommand    ( CmdID          : LongWord;     // = 1 (to pull the data servers list)
                    const CmdStr   : WideString;   // = <Host name>;<ClientID>
                                                   Example: CmdStr = 'Host1;1277608648'
                    CmdInfo        : OleVariant;
                    var CmdResult  : OleVariant); safecall
                                   ;               //A two dimensional array with the data
                                                   servers information
```

Description: This additional object also implements the interface IMetilinx, which is accessible through the function.

```
ProcessCommand    ( CmdID          : LongWord;     // = 2 (to pull the connection string list)
                    const CmdStr   : WideString;   // = <Host name>;<ClientID>
                    CmdInfo        : OleVariant;   //
                    var CmdResult  : OleVariant); safecall
                                   ;               // A two dimensional array with the data
                                                   servers information
```

Description: This additional object also implements the interface IMetilinx, which is accessible through the function.

```
ProcessCommand    ( CmdID          : LongWord;     // = 3 (to open an ADO connection using a
                                                   connection ID)
                    const CmdStr   : WideString;   // <ConnectionID>
                    CmdInfo        : OleVariant;
                    var CmdResult  : OleVariant);  // A two dimensional array containing the
                                   safecall;       ADO connection object and a
                                                   connection-opened-ID.
```

Description: This additional object also implements the interface IMetilinx, which is accessible through the function.

```
ProcessCommand    ( CmdID          : LongWord;     // = 4 (close a currently open connection)
                    const CmdStr   : WideString;   // = <ConnectionOpenedID>
                    CmdInfo        : OleVariant;   //
                    var CmdResult  : OleVariant);  // = No values returned
                                   safecall;
```

Description: This additional object also implements the interface IMetilinx, which is accessible through the function.

Note:

Please observe the parameter differences when using different CmdID values.

IMetilinx Interface

Below are the interface specifications for the COM object IMetiLinx. Also included is the ProcessCommand function needed for Marshalling Scripts and Generic Scripts.

```
//
*****************************************************************//
// Interface: IMetilinx
// Flags:     (320) Dual OleAutomation
// GUID:      {D1FACAFC-C509-11D3-8775-0050046EDE16}
//
*****************************************************************//
IMetilinx = interface(IUnknown)
        ['{D1FACAFC-C509-11D3-8775-0050046EDE16}'] procedure ProcessCommand(CmdID: LongWord;
        const CmdStr: WideString;
        CmdInfo: OleVariant; var CmdResult:
OleVariant); safecall; end;

//
*****************************************************************//
```

```
Function ProcessCommand(CmdID, CmdStr, CmdInfo)
CmdID, CmdStr and CmdInfo  are Variants
//
*****************************************************************//
```

Additional Topics:

Data Load Object (DLO)

Publishing Host Information

Data Load Object (DLO)

DLO is a COM object contained in the `Dlo.exe` file supplied with version 2.2, which requires you to develop your own COM interface. This object provides information about data servers and connections associated with a host. It supports user's business rules implementation by enabling more dynamic opening of database connections.

By creating your own interface, you can use the DLO to replicate objects to databases and recover failed database connections. The DLO enhances your use of the iManager optimization mechanism by providing a sorted list of connection strings based on server-load statistics. It implements the IDataLoadObject and IMetiLinx interfaces.

Data Load Object (DLO)

DLO is a COM object contained in the `Dlo.exe` file supplied with version 2.2, which requires you to develop your own COM interface. This object provides information about data servers and connections associated with a host. It supports user's business rules implementation by enabling more dynamic opening of database connections.

By creating your own interface, you can use the DLO to replicate objects to databases and recover failed database connections. The DLO enhances your use of the iManager optimization mechanism by providing a sorted list of connection strings based on server-load statistics. It implements the IDataLoadObject and IMetiLinx interfaces.

Information Messages

Message ID	**imGetSQLInfo**
Message Number	$00000101
Explanation	This message is used to perform a standard SQL query like SELECT, to get a small amount of data from a dataset.
Required Payload	SQL query statement.
Payload Return	The data requested with each field separated by TAB and each record separated by carriage return + line feed (CRLF).
Code Example (PASCAL)	BufferStr := 'Select GetDate()'; StrPCopy(Buffer , BufferStr); MsgID:= imGetSQLInfo; SendMsg (MsgID, ClientID, Buffer); CurrentDT := StrToDateTime(Copy(Buffer,1, Pos(#9,Buffer) -1));
Result	The above example returns the date and time value on the server. ClientID is the value returned by the Login function.
Function Word	SendMsg

Message ID	**imGetStoredProcedure**
Message Number	$00000102
Explanation	This message calls a stored procedure that returns a dataset.
Required Payload	Name of the stored procedure and the parameters required by the stored procedure. All of this information should be separated by CRLF.
Payload Return	The result data returned by the stored procedure with each field separated by TAB and each record separated by CRLF.
Code Example (PASCAL)	BufferStr := StoredProcedureName + #13#10 + spParameter1 + #13#10 + spParameter2... StrPCopy (Buffer , BufferStr); MsgID:=imGetStoredProcedure; SendMsg (MsgID , ClientID , Buffer);
Result	The example will call "StoredProcedureName" with parameters "spParameter1" and "spParameter2" and retrieves the result data returned by the stored procedure in the "Buffer," with each field separated by TAB and each record separated by CRLF.
Function Word	SendMsg

Message ID	**imExecStoredProcedure**
Message Number	$00000103
Explanation	This message calls a stored procedure that does not return a dataset.
Required Payload	Name of the stored procedure and the parameters required by the stored procedure. All of this information should be separated by CRLF.

Payload Return	A list of returning parameters with each field separated by TAB and ending in CRLF.
Code Example (PASCAL)	BufferStr := StoredProcedureName + #13#10 + spParameter1 + #13#10 + spParameter2... StrPCopy (Buffer , BufferStr); MsgID:=imExecStoredProcedure; SendMsg (MsgID , ClientID , Buffer);
Result	The example will call "StoredProcedureName" with parameters "spParameter1" and "spParameter2" and retrieves the result data returned by the stored procedure in the "Buffer," with each field separated by TAB and each record separated by CRLF.
Function Word	SendMsg

Message ID	**imGetFile**
Message Number	$00000104
Explanation	This message transfers files specified by the Host application from the Update directory on the Host server to the client PC.
Required Payload	Origin Filename to be transferred from the Host, as well as the destination path and Filename. The source and the destination values should be separated by a CRLF. The destination path must be valid for the local client machine.
Payload Return	No return value.
Code Example (PASCAL)	BufferStr := 'Readme.txt' + #13#10 + 'C:\mydir\Readme.txt'; StrPCopy(Buffer, BufferStr); MsgID:=imGetFile; SendMsg(MsgID, ClientID, Buffer) ;
Result	The above example will transfer file "Readme.txt" from the Update Directory on the Host server to the clients' local machine in the folder "c:\mydir". ClientID is the value returned by the Login function.
Function Word	SendMsg

Message ID	**imGetFileDetails**
Message Number	$00000110
Explanation	This message is used to get the name, size and date stamp of the file(s) in the Update directory of the Host system. This Update directory is created during the creation of the Host system and is located in the deployment folder of the Host system you are working with.
Required Payload	No value required
Payload Return	The corresponding information for all existing files in the Update folder Data values for a single file are TAB-separated. TAB and CRLF separate entries for different files. (Filename#9Filesizein bytes#9Filedate#9Filetime#9#13#10)
Code Example (PASCAL)	StrPCopy(Buffer , #0); MsgID:=imGetFileDetails; SendMsg (MsgID, ClientID, Buffer);

Result	In the above example, the buffer will contain all information for the file, in the specified directory, on the server, in the format explained above. ClientID is the value returned by the Login function.
Function Word	SendMsg

Message ID	**imGetBinaryInfo**
Message Number	$00000120
Explanation	This message is used to retrieve query results like SELECT in binary format. It is especially useful to get Blobs and image fields.
Required Payload	SQL query statement
Payload Return	Pointer to a buffer containing the query result in binary format and the size of the returned buffer in bytes.
Code Example (PASCAL)	BufferStr := 'Select BMP From Animals Where Name = ''Boa'''; StrPCopy(Buffer , BufferStr); BufferSize:=strLen(Buffer); MsgID:=imGetBinaryInfo; SendMsgB(MsgID , ClientID , Buffer ,BufferSize);
Result	The above example will get a blob field from the table "Animals". The size of the blob is returned in 'BufferSize'.
Function Word	SendMsgB

Information Messages

Message ID	**imGetSQLInfo**
Message Number	$00000101
Explanation	This message is used to perform a standard SQL query like SELECT, to get a small amount of data from a dataset.
Required Payload	SQL query statement.
Payload Return	The data requested with each field separated by TAB and each record separated by carriage return + line feed (CRLF).
Code Example (PASCAL)	BufferStr := 'Select GetDate()'; StrPCopy(Buffer , BufferStr); MsgID:= imGetSQLInfo; SendMsg (MsgID, ClientID, Buffer); CurrentDT := StrToDateTime(Copy(Buffer,1, Pos(#9,Buffer) -1));
Result	The above example returns the date and time value on the server. ClientID is the value returned by the Login function.
Function Word	SendMsg

Message ID	**imGetStoredProcedure**
Message Number	$00000102
Explanation	This message calls a stored procedure that returns a dataset.
Required Payload	Name of the stored procedure and the parameters required by the stored procedure. All of this information should be separated by CRLF.
Payload Return	The result data returned by the stored procedure with each field separated by TAB and each record separated by CRLF.
Code Example (PASCAL)	BufferStr := StoredProcedureName + #13#10 + spParameter1 + #13#10 + spParameter2... StrPCopy (Buffer , BufferStr); MsgID:=imGetStoredProcedure; SendMsg (MsgID , ClientID , Buffer);
Result	The example will call "StoredProcedureName" with parameters "spParameter1" and "spParameter2" and retrieves the result data returned by the stored procedure in the "Buffer," with each field separated by TAB and each record separated by CRLF.
Function Word	SendMsg

Message ID	**imExecStoredProcedure**
Message Number	$00000103
Explanation	This message calls a stored procedure that does not return a dataset.
Required Payload	Name of the stored procedure and the parameters required by the stored procedure. All of this information should be separated by CRLF.

Payload Return	A list of returning parameters with each field separated by TAB and ending in CRLF.
Code Example (PASCAL)	BufferStr := StoredProcedureName + #13#10 + spParameter1 + #13#10 + spParameter2... StrPCopy (Buffer , BufferStr); MsgID:=imExecStoredProcedure; SendMsg (MsgID , ClientID , Buffer);
Result	The example will call "StoredProcedureName" with parameters "spParameter1" and "spParameter2" and retrieves the result data returned by the stored procedure in the "Buffer," with each field separated by TAB and each record separated by CRLF.
Function Word	SendMsg

Message ID	**imGetFile**
Message Number	$00000104
Explanation	This message transfers files specified by the Host application from the Update directory on the Host server to the client PC.
Required Payload	Origin Filename to be transferred from the Host, as well as the destination path and Filename. The source and the destination values should be separated by a CRLF. The destination path must be valid for the local client machine.
Payload Return	No return value.
Code Example (PASCAL)	BufferStr := 'Readme.txt' + #13#10 + 'C:\mydir\Readme.txt'; StrPCopy(Buffer, BufferStr); MsgID:=imGetFile; SendMsg(MsgID, ClientID, Buffer) ;
Result	The above example will transfer file "Readme.txt" from the Update Directory on the Host server to the clients' local machine in the folder "c:\mydir". ClientID is the value returned by the Login function.
Function Word	SendMsg

Message ID	**imGetFileDetails**
Message Number	$00000110
Explanation	This message is used to get the name, size and date stamp of the file(s) in the Update directory of the Host system. This Update directory is created during the creation of the Host system and is located in the deployment folder of the Host system you are working with.
Required Payload	No value required
Payload Return	The corresponding information for all existing files in the Update folder Data values for a single file are TAB-separated. TAB and CRLF separate entries for different files. (Filename#9Filesizein bytes#9Filedate#9Filetime#9#13#10)
Code Example (PASCAL)	StrPCopy(Buffer , #0); MsgID:=imGetFileDetails; SendMsg (MsgID, ClientID, Buffer);

Result	In the above example, the buffer will contain all information for the file, in the specified directory, on the server, in the format explained above. ClientID is the value returned by the Login function.
Function Word	SendMsg

Message ID	**imGetBinaryInfo**
Message Number	$00000120
Explanation	This message is used to retrieve query results like SELECT in binary format. It is especially useful to get Blobs and image fields.
Required Payload	SQL query statement
Payload Return	Pointer to a buffer containing the query result in binary format and the size of the returned buffer in bytes.
Code Example (PASCAL)	BufferStr := 'Select BMP From Animals Where Name = ''Boa'''; StrPCopy(Buffer , BufferStr); BufferSize:=strLen(Buffer); MsgID:=imGetBinaryInfo; SendMsgB(MsgID , ClientID , Buffer ,BufferSize);
Result	The above example will get a blob field from the table "Animals". The size of the blob is returned in 'BufferSize'.
Function Word	SendMsgB

Code Sample

The example included here shows how to use `ioCOMRemote.dll` for executing your remote COM objects and for querying the DLO object at the server. The sample also shows how to connect to a host and invoke the DLO methods using a host message.

Note:

You need to import all of your COM objects using the IMetiLinx interface. If you wish to obtain a list of data servers for a given host, import the DLO object into iManager.

```
Dim Obj As Object

Private Sub Command1_Click()

 Caption = "Connecting..."
 Set Obj = CreateObject("ioComRemote.MetilinxClient")
 Call Obj.Connect("HOST1", "192.168.2.84", 1024, "username1", "password", "metilinx", status)
 Caption = "Connected to 192.168.2.84"
End Sub

Private Sub Command2_Click()
 Dim msgid   As Variant
 Dim res     As Variant
 Dim res1    As Variant
 Dim command As Variant
 Dim status  As Variant

 'executing a COM object that resides in the server side
 msgid = 2147483650# 'the first message after the last iManager reserved message
                     'corresponding to hex number 80000002
 command = "select * from authors"
 Call obj.Execute(msgid, command, res, status)
 Label1.Caption = "value >>>> " + res

 msgid = 2147483651# 'the second message after the last iManager reserved message
                     'corresponding to hex number 80000003
 command = "HOST1;" + "2400380870"
 Call obj.Execute(msgid, command, res, status) 'to obtain the data servers
 Label1.Caption = "DataServer 1>>>> " + res(0, 0) + " STATISTIC=" + res(0, 1)
 Call obj.Execute(msgid + 1, command, res1, status) 'to obtain the connection strings
 Label2.Caption = "ConnStr 1>>>> ID=" + res1(0, 0) + " CONNSTR=" + res1(0, 1)
 Set res = Nothing
 Set res1 = Nothing
 End Sub

Private Sub Command3_Click()
 Set obj = Nothing
 Unload Me
End Sub
```

```
Private Sub Form_Load()
  Caption = "Client using ioCOMRemote.DLL"
End Sub
```

Additional Topics:

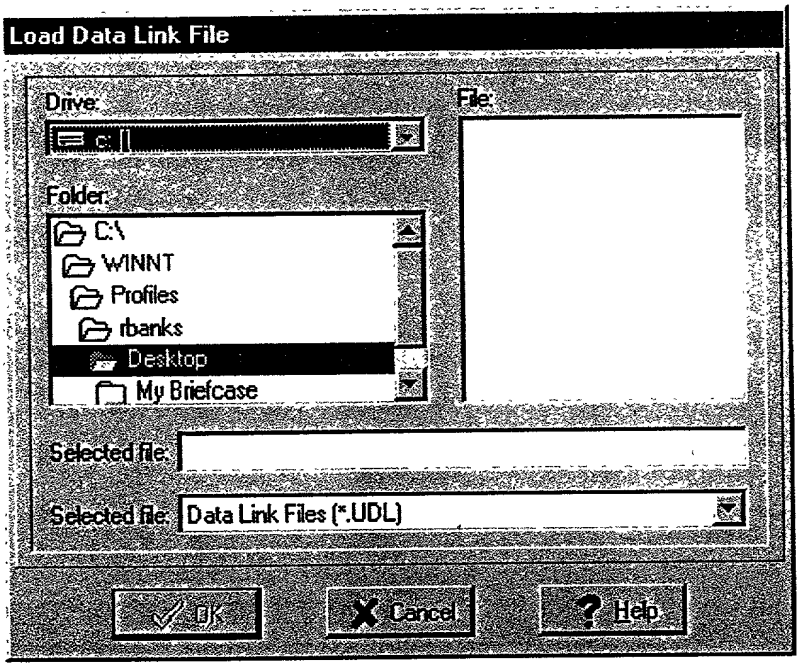
Data Load Object

ioCOMRemote

Load Data Link File

If you are specifying a Microsoft Data Link connection, you must load an existing data link (.udl) file

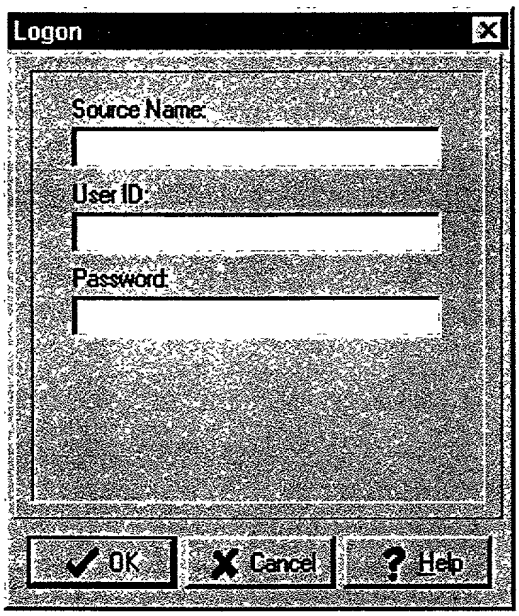
Drive	Select drive on which the .UDL file resides.
Folder	Select folder on which the .UDL file resides.
Selected File	Path of the .UDL file.
Selected File	Select Data Link Files.



Load Data Link dialog box

Logon

Source Name	The Source ID located in the Host detail panel.
User ID	User login ID.
Password	Password of User ID



Logon dialog box

Member Descriptor

A member descriptor is a five-element array of variant that describes a member (function or property) of an object, including parameters and result, if needed. Use it to specify member calls and marshal results when remotely invoking.

MemberDescriptor: Variant containing a one dimensional array of Variant with 5 elements where:

MemberDescriptor [0] : Name of the interface the member belongs to. If the member name is a nested reference using dot notation, this element refers to the interface containing the first property from the left.

MemberDescriptor [1] : Name of the member to invoke. Nested references to members (using dot notation) allowed.

MemberDescriptor [2] : Flags describing the invocation context. as follows:

FLAG NAME	FLAG VALUE	DESCRIPTION
INVOKE_FUNC	1	The member is invoked as a method.
INVOKE_PROPERTYGET	2	The member is retrieved as a property or data member
INVOKE_PROPERTYPUT	4	The member is set as a property or data member
INVOKE_PROPERTYPUTREF	8	The member is set by a reference assignment, rather than a value assignment. This flag is valid only when the property accepts a reference to an object.

MemberDescriptor [3] : ParamLst, one dimensional array of Variant with 4 elements describing the parameter list.

- ParamLst[0] : Number of parameters the member takes.
- ParamLst[1] : Numer of named parameters
- ParamLst[2] : VarArray with as many elements as parameters the member takes. Parameter matching is made from left to right. Values for input parameters must be set prior invocation. Type matching between array elements and parameters is responsibility of the caller.
- ParamLst[3] : Array of named parameters. See examples of how to define the Parameter List.

MemberDescriptor [4] : Variant where the result is marshaled back to the caller, or NULL if the caller expects no result. This argument is ignored if INVOKE_PROPERTYPUT or INVOKE_PROPERTYPUTREF is specified.

MetiLinx iManager Messages

According to their functionality and from the client application's viewpoint, message identifiers can be separated into three different categories: Request Messages, Answer Messages and Internal Messages. Whenever the requested action is performed, the Host sends an answer message ID (acknowledge message). If errors arise, the corresponding message IDs are sent back.

Request Messages

Request Messages are sent from the client to the Host. They represent actions the Host system can perform and send results or answers back to the client application. There are two kinds of request messages: Stateless Messages and Transaction (State-based) Messages.

Stateless Messages

After executing a Stateless Message, the Host sends the corresponding reply message back without keeping any reference to it. Stateless Messages include update messages and single-shot information requests, as described below.

Transaction Messages (State-based)

A Transaction Message is comprised by messages associated to "states" of the Host. For example, requests to open a table and retrieve its content page by page belong to this class. Depending on its current state, requests to the Host will be accepted or declined. For example, starting a transaction on the database server requires the client close it (by commitment or rollback). In a different situation, after opening a dataset, the client can browse through it, but it must be closed before opening a new one. This is the meaning of *"states of the Host"*.

Answer Messages

Answer messages are the replies to the client's requests from the Host. Usually, the Host combines more than one message identifier to give additional information about its state and/or to report errors detected while performing the requested action. For example, when replying to a single-shot message, the Host might return **imDone**, meaning the request was carried out successfully and no other information is available. Alternately, it can combine an error message identifier with a reply message identifier to indicate that although information was sent to the client, not all information could be accommodated into the buffer, producing an overflow situation. The combination of message identifiers is done using a logical **OR** operation, so that the **AND** operation has to be used in order to detect the presence of a message identifier in a reply message.

In case of multiple-shot messages, the Answer message identifiers can inform the client about the cursor position inside the dataset, data errors, etc. This facilitates the process of browsing through the dataset and detecting when an endpoint is reached.

Whenever imDone is returned (alone or combined with some other message identifiers), it means the Host was able to obtain data and send it to the client. Otherwise, only the corresponding error message identifiers will be sent back to the client, indicating that an error was raised during the request execution and the Host could not complete the task. If more information is available, it will be sent back as a payload.

Report Answer Messages

There are other Answer messages the client application might encounter that originate either at the Host or in the ioRemote Object. These are helpful in the debugging of applications during development.

Internal Messages

Internal Messages are used in the communication between the ioRemote Object and the Host to perform operations related to client's requests. These messages are not sent to or received by the client; therefore, they will not be listed in this document.

Setup Files

To download MetiLinx iManager Developer 2.2 from the MetiLinx Web site, you must register and agree to the License Terms of the Software License Agreement MetiLinx, Inc. grants. Download the file, `MetiLinxEnterprise2.2.exe`, to the Download directory.

Tip:

Be sure to carefully read the License Terms of the Software License Agreement. MetiLinx, Inc. grants the following licenses for the use of MetiLinx iManager Developer 2.2:

- 30-Day Evaluation
- Development
- Enterprise

Evaluation use of the software begins upon downloading the software and precisely ends 30 days, thereafter.

To Uninstall MetiLinx iManager Developer 2.2

1. On the **Start** menu, point to **Settings**, and then select **Control Panel**.
2. Double-click on the **Add/Remove Programs** icon.
3. Click the **Install/Uninstall** tab.
4. From the list of programs that Windows can remove, select **iManager 2.2**.
5. Click **Add/Remove**.
6. At the prompt, click **Yes** to confirm that you want to remove the MetiLinx Enterprise 2.2 program.

Note:

You may safely respond Yes to All when the uninstall program prompts you to confirm the removal of the following files located in **C:\ProgramFiles\Common \MetiLinx\MetiLinx Enterprise 2.2**:

Procdata.dll

Sysdata.dll

MetilinxObject.dll

Msscript.ocx

QueryObject.exe

Microsoft OLE DB provider table

Provider Name	Data Source	Provider	Product
SQLOLEDB	SQL Server	Microsoft OLE DB Provider for SQL Server	SQL Server
MSDAORA	Oracle	Microsoft OLE DB Provider for Oracle	Any (2)
Microsoft.Jet.OLEDB.4.0	Access/Jet	Microsoft OLE DB Provider for Jet	Any
MSDASQL	ODBC data source	Microsoft OLE DB Provider for ODBC	Any
MSIDXS	File system	Microsoft OLE DB Provider for Indexing Service	Any
Microsoft.Jet.OLEDB.4.0	Microsoft Excel Spreadsheet	Microsoft OLE DB Provider for Jet	Any

Microsoft OLE DB provider table

Provider Name	Data Source	Provider	Product
SQLOLEDB	SQL Server	Microsoft OLE DB Provider for SQL Server	SQL Server
MSDAORA	Oracle	Microsoft OLE DB Provider for Oracle	Any (2)
Microsoft.Jet.OLEDB.4.0	Access/Jet	Microsoft OLE DB Provider for Jet	Any
MSDASQL	ODBC data source	Microsoft OLE DB Provider for ODBC	Any
MSIDXS	File system	Microsoft OLE DB Provider for Indexing Service	Any
Microsoft.Jet.OLEDB.4.0	Microsoft Excel Spreadsheet	Microsoft OLE DB Provider for Jet	Any

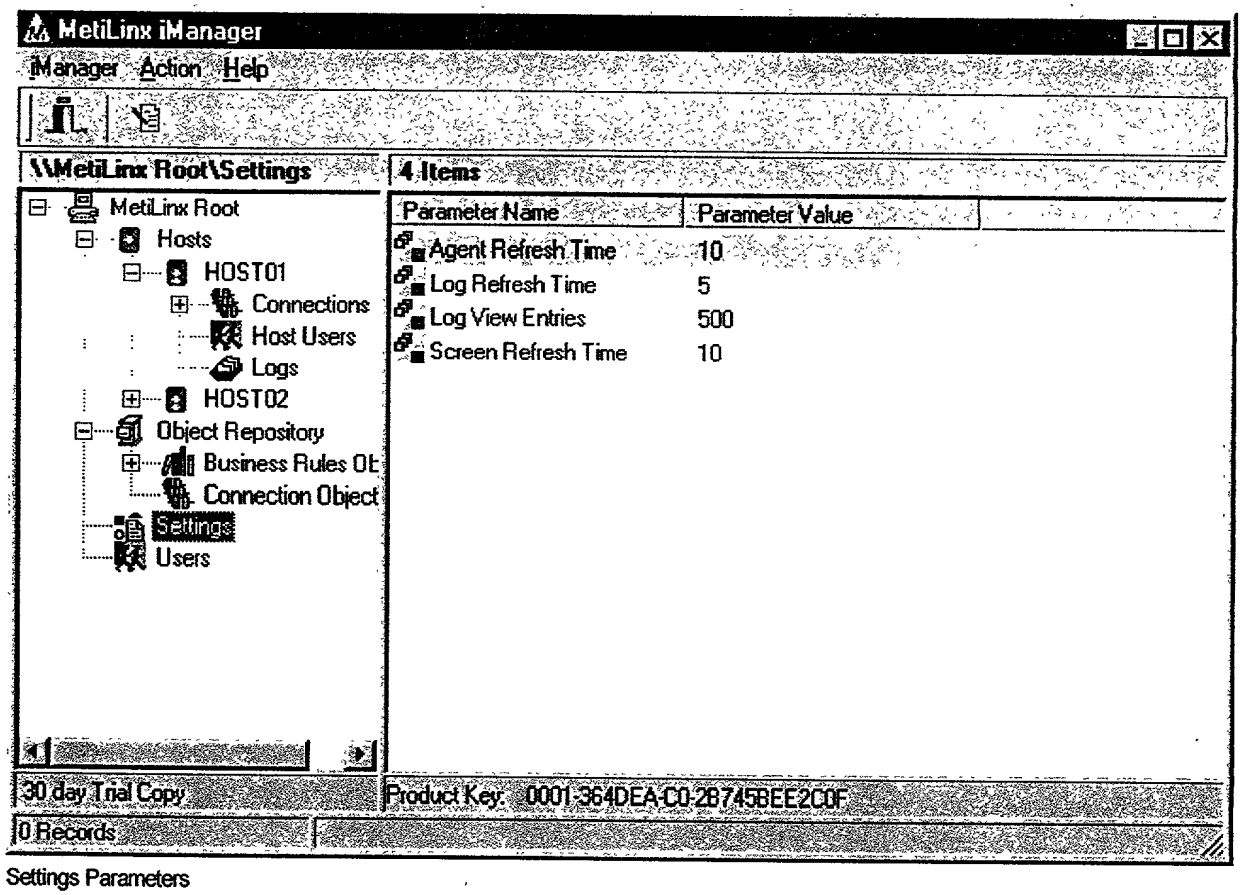
Modify Connection Object

Change the status or connection string of a connection object in the Object Repository.

Connection Object Status	Select Active to activate the connection object. Select Inactive to pause the connection. If iManager detects a connection failure with a connection object, the status of the connection object automatically changes to Inactive. Every 15 minutes, iManager attempts to reconnect the connection. If the connection object reconnects, iManager changes the status to Active. Select Out of Service to permanently place an object connection offline. If a connection fails to reconnect, consider placing it out of service.
Connection String	Select to modify the connection string parameters of the connection object.

Modify Settings Parameters

Agent Refresh Time	Use this parameter to adjust, in seconds, how often iAgent checks the Host status and activates its data publishing mechanism. Minimum Value: 5 Maximum Value: 3600 Recommended Value: 10
Log Refresh Time	Use this parameter to adjust, in seconds, how often iManager retrieves messages from the administration database when logging is enabled. Minimum Value: 1 Maximum Value: 100 Recommended Value: 5
Log View Entries	Use this parameter to adjust the number of message displayed in the Log window. Minimum Value: 100 Maximum Value: 5000 Recommended Value: 500
Screen Refresh Time	Use this parameter to adjust, in seconds, how often the Host Status information in the Hosts window updates. Minimum Value: 3 Maximum Value: 100 Recommended Value: 10



Settings Parameters

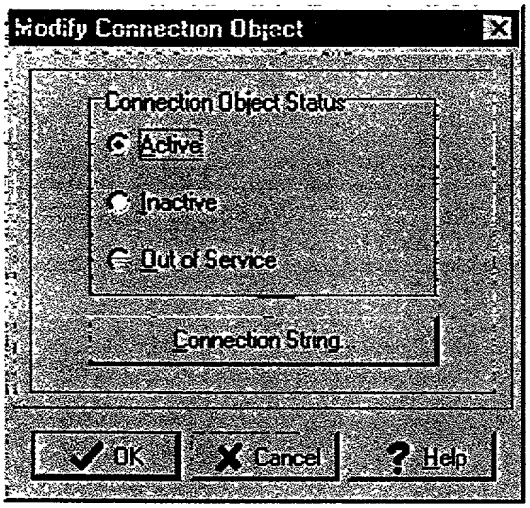
Note:

Decreasing the parameters to the lowest settings is not recommended, as doing so will slow down the Host performance.

Modifying a Connection Object

1. At the **MetiLinx** tree, click **Object Repository**; then click **Connection Objects**.
2. In the right pane, right-click the **Connection Object** you wish to modify; then click **Modify Connection Object**.

iManager opens the **Modify Connection Object** window.



3. If you want to, modify the Connection Object Status,
4. Click **Connection String**.
5. Click **OK** to implement the changes.

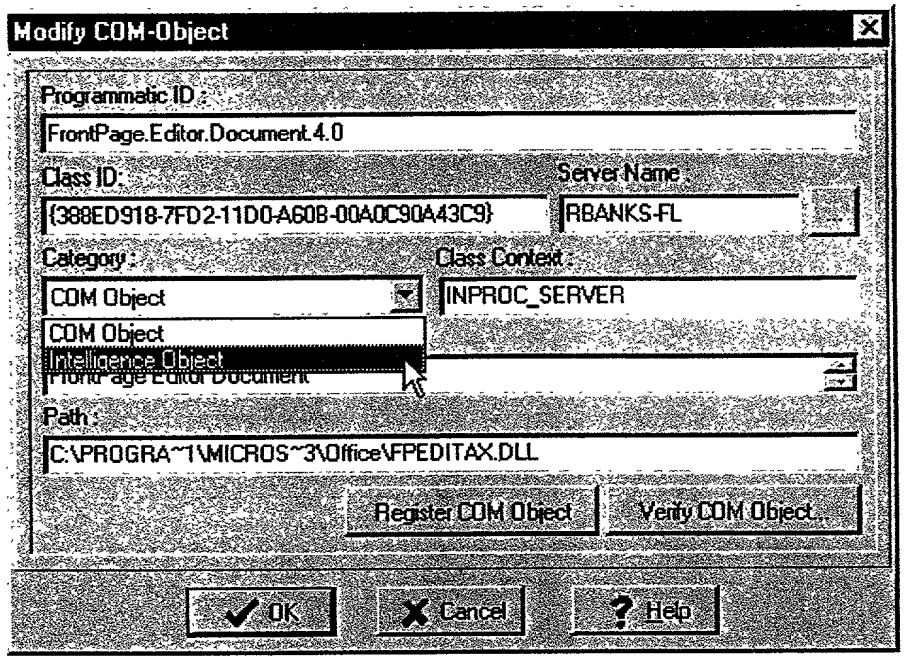
Modifying a Script

1. Select the script you want to modify located in **Object Repository > Business Rules Objects > Scripts**.
2. Click **Action | Modify Script**.
3. Make the changes and click **OK** to implement them.

Modifying an Object

1. Click the COM Object you want to modify found in **Object Repository >Business Rules Objects >COM Objects.**
2. Click **Action | Modify Object**.

iManager opens the **Modify COM-Object** dialog box.



There are two available tools on the "Modify COM-Object" screen:

Verify COM Object	This feature will test your object properties for accuracy.
Register COM Object	This feature will register your object with the local System.

3. Modify the object and click **OK** to save your changes.

Additional Topics:

Verifying COM objects

To Uninstall MetiLinx iManager Developer 2.2

1. On the **Start** menu, point to **Settings**, and then select **Control Panel**.
2. Double-click on the **Add/Remove Programs** icon.
3. Click the **Install/Uninstall** tab.
4. From the list of programs that Windows can remove, select **iManager 2.2**.
5. Click **Add/Remove**.
6. At the prompt, click **Yes** to confirm that you want to remove the MetiLinx Enterprise 2.2 program.

Note:

You may safely respond Yes to All when the uninstall program prompts you to confirm the removal of the following files located in **C:\ProgramFiles\Common \MetiLinx\MetiLinx Enterprise 2.2**:

Procdata.dll

Sysdata.dll

MetilinxObject.dll

Msscript.ocx

QueryObject.exe